UYMS 2019
Turkish National Software Engineering Symposium

Proceedings of the 13th Turkish National Software Engineering Symposium

İzmir, Turkey, September 23-25, 2019.

Edited by

Tolga Ayav *
Murat Erten **
Selma Tekir *

* Izmir Institute of Technology, Department of Computer Engineering, Izmir, Turkey
** University of Bakırçay, Department of Computer Engineering, Izmir, Turkey

---

Table of Contents

- Önsöz (Foreword)

Session 1: Davetli Konuşma

- Touching an Intelligent World
  Lina Cao

Session 2A: Yazılım Endüstrisi

- Yazılım Endüstrisi ve Akademi arasındaki Boşluğu Doldurmaya Yönelik bir Anket Çalışması: İlk Sonuçlar (A Survey on Bridging the Gap between Software Industry and Academia: Preliminary Results)
  Deniz Akdur
- Türkiye'de Yazılım Geliştirme Takımlarının Bulut Kullanımı, Kullanım İsteği ve Uygunluk Algısı (Actual Use, Use Intention and Perceived Suitability of Cloud by Turkish Software Development Teams)
  Erhan Pişirir, Cüneyt Sevgi, Oumout Chouseinoglou, Erkan Uçar

Session 2B: Yazılım Uygulamaları

- Konferans, Kongre ve Sempozyum Yönetim Sistemleri İçin Bir Mobil İstemci Uygulama (A Mobile Client Application for Conference, Congress and Symposium Managements Systems)
  Serhat Celil İleri, Selçuk Aslan, Sercan Demirci
- Nesnelerin İnterneti için Kural Denetleme Mekanizması (Rule Verification Mechanism for Internet of Things)
  Cengiz Togay, Faik Özgür, Gizem Kiraz, Durmuş Kurtuluş

Session 2C: Yazılım Test Mühendisliği-1

- Testte Çevresel Değişikliklerin Denetiminde Olay-Yönlendirmeli İzleme Yaklaşımı
(Testte Çevresel Değişikliklerin Denetiminde Olay-Yönlendirmeli İzleme Yaklaşımı)
Şükrü Alataş, Aysu Betin-Can
- Yürütülebilir Doğal-Dil Test Gereksinimler: Bir Test-Otomasyon Deneyim Raporu
(Executable Natural-Language Test Specifications: A Test-Automation Experience Report)
Vahid Garousi, Alper Buğra Keleş, Zeynep Özdemir Güler, Yunus Balaman

Session 3A: Akıllı Yazılım Uygulamaları

- Derin Öğrenme Modellerinde İzge Tabanlı Hata Konumlandırma
(Spectrum Based Fault Localization for Deep Learning Models)
Mehmet Melih Arıcı, Hasan Ferit Enişer, Alper Şen
- Aşırı Öğrenme Makinesi Algoritmasını Kullanarak Yazılım Zafiyet Kestirimi
(Software Vulnerability Prediction using Extreme Learning Machines Algorithm)
Ali Seydi Keçeli, Aydın Kaya, Çağatay Çatal, Bedir Tekinerdoğan
- Yapay Zeka Tabanlı Yüz Tanıma Sisteminin Geliştirilmesi Ve Optimizasyonu
(Development and Optimization of Artificial Intelligence Based Face Recognition System)
Sedat Yıldız, Erdem Özgür, Nihal Bilal

Session 3B: Yazılım Ölçme Altyapıları

- Yazılım Geliştirme Sektöründe Kurumsal Ölçümlerin Belirlenmesi ve Ölçüm Altyapısının Oluşturulması
(Determination of Corporate Metrics in Software Development Sector and Establishment of Measurement Infrastructure)
Melike Turgut, Fatma Gül Bulut, Haluk Altunel
- Türkiye Ulusal Yazılım Mühendisliği Sempozyumu Bildirilerinin Konu Analizi
(Topic Analysis of Turkish National Software Engineering Symposium Papers)
Görkem Giray
- Web Erişilebilirlik Değerlendirmesi:Sistematik Haritalama
(Web Accessibility Evaluation: Systematic Mapping)
Ömer Naci Soydemir, Pınar Onay Durdu

Session 3C: Yazılım Test Mühendisliği-2

- Çalışma Sırasında Konfigüre Edilebilir Ortak Gömülü Simülatör
(Run-Time Configurable Common Embedded Simulator)
Erkan Tan, İbrahim Balci, Uğur Yilmaz, Elif ÖztÜrk, Derya Yeliz UlutaŞ
- Yazılım Ürün Hatları için Özellik Yönelimli Test Modellerinin Yönetimi
(Management of Feature-Oriented Test Models for Software Product Lines)
Tugkan Tuglular, Mutlu Beyazıt, Dilek Ozturk
- Pratikte Model Tabanlı Test: Bankacılık Alanından Bir Deneyim Raporu
(Model-based testing in practice: An experience report from the banking domain)
Şerafettin Şentürk, Abdurrahman Akın, Ayşe Betül Karagöz, Vahid Garousi

Session 3D: Yazılım Modelleme

- Aygıt Ağacı Yazılımlarının Model Güdümlü Geliştirilmesinin Tersine Mühendislik ile Desteklenmesi
(Reverse Engineering Support for the Model-driven Development of Device Tree Software)
Sadık Arslan, Geylani Kardas
- Meta-Modelleme Araçlarının Uygulayıcıların İhtiyaçları Açısından Analizi
(Towards Analysing the Meta-Modeling Tools for the Needs of Practitioners)
Mert Ozkaya, Deniz Akdur
- Bir Yazılım Sisteminin Kullanımının Süreç Madenciliği Teknikleri ile Modellenmesi: Endüstriyel Bir Vaka Çalışması
(Modelling the Use of a Software System with Process Mining Techniques: An Industrial Case Study)
Umit Ulkem Yildirim, Beytullah Sarıca, Ethem Utku Aktas, Mehmet Calpur

Session 4A: Yazılım Uygulamaları

- Yazılım Geliştirme Dış Kaynak Alımlarında Start-Up Firmalar için Yeni bir Satın Alma Modeli
(A New Buying Model For Startup Companies At Software Development Outsourcing Procurement)
Selin Çalışkan, Özlem Özenç, Şerife Çağla Özgöç, Okan Horuz
- Yazılım Yaşam Döngüsü Modelleri için Sistematik Eşleme Çalışması
(Systematic Mapping Study for Software Lifecycle Models)
Yagup Macit, Ahsen İkbal Aytekin, Çağdaş Üsfekes
- Model Tabanlı Geliştirme Teknolojisi Kullanılarak Geliştirilen Hava Aracı Yazılımlarının Sertifikasyon Sürecine Yönelik Öneriler

(Suggestions for the Certification Process of Air-borne Software Developed Using Model-Based Development Technology)
TuĞba SaraÇ

Session 4B: Akıllı Yazılım Uygulamaları-2

- Gri Kurt Optimizasyonu (GWO) Algoritması İle Finans Sektörü İçin Kredi Verilebilirlik ve Sınıflandırma Çalışması
(Creditability and Classification Work for Finance Sector with Gray Wolf Optimization (GWO) Algorithm)
Ali Tunç
- Makine Öğrenmesi Sistemi Geliştirmek İçin Yazılım Sürecini Uyarlamadaki Zorluklar: Bir Vaka Çalışması
(Challenges in Adapting Software Process for Developing a Machine Learning System: A Case Study)
Görkem Giray, Murat Osman Ünalır, Sena Koçer

Session 4C: Yazılım Test Mühendisliği-3

- Bir Ölçüm Cihazının Programlama Arayüzü için Model Öğrenme
(Model Learning for the Programming Interface of a Measurement Device)
Atakan Garipler, Ege Sorguç, Evrim Özel, Salih Emre Uslu, Mahdi Saeedi Nikoo, Halit Oğuztüzün
- Hata Verisi Analizi Ve Değişkenlerin Girdi Uzayının Ayrıklaştırılmasını Kullanan Bir Birleşimsel Test Verisi Üretimi Yaklaşımı
(A Combinatorial Test Data Generation Approach Using Fault Data Analysis And Discretization Of Parameter Input Space)
Hakan Bosnalı

Session 4D: Sağlık için Yazılım Mühendisliği-1

- EEG Sinyallerinin Analizi ve Sınıflandırılmasında Yazılım Sistemlerinin Kullanımına Genel Bir Bakış
(A General Approach to Usage of Software Systems in EEG Signal Analysis and Classification)
Buse Nur Uygun, Emine Sezer, Murat Osman Ünalır
- Laboratuvar Bilgi Sistemlerinin Bulut Dönüşümü
("Laboratory Information System" Transfer to Cloud)
Bengi Tuğcu İdemen, Arhun Hınçalan

Session 5A: Yazılım Süreç Yönetimi

- Çevik Ontoloji Geliştirme 101: Yeniden Kullanım Tabanlı Çevik Bir Ontoloji Geliştirme Yöntemi
(Agile Ontology Development 101: A Reuse-based Agile Ontology Development Methodology)
Cemil Abiş, Görkem Giray, Murat Osman Ünalir
- Otomotiv Sektöründe Genişletilmiş Çevik Yazılım Geliştirme Süreci Deneyimi
(Scaled Agile Software Development Process Experience in Automotive)
Fatih Öztürk, Koray Bilgin, Cem Sezer, Seda Gunes
- Çevik Yöntemlerde Büyüklük Ölçümü: Bağımsız Ölçümler Mümkün Mü?
(Size Measurement in Agile Methods: Are Independent Measurements Possible?)
Tuna Hacaloglu, Aylin Deveci, Selami Bağrıyanık, Onur Demirors

Session 5B: Veri Analizi ve Yönetimi

- Heterojen Veritabanı Yönetim Sistemleri için Bir Tekil Veri Erişim Çerçevesi
(A Single Data Access Framework for Heterogeneous Database Management Systems)
Nail Diker, Görkem Giray, Murat Osman Ünalir
- Çoklu Veri Saklama Altyapısı
(Polyglot Persistence Infrastructure)
Cahit Yiğit Çetiner, Merve Kaymak, Mustafa Karabacak, Tuba Yağlı
- Yazılım Mühendisliğinde Büyük Veri Analizi: MEASURE – WEKA
(Big Data Analysis in Software Engineering: MEASURE – WEKA Integration)
Utku Apaydin, Selami Bagriyanik, Adem Karahoca, Mete Aktaş, Tolga Ensari, Sinan Verdi, Ayşe Tosun, Hamit Doğan

Session 5C: Yazılım Süreç Yönetimi

- Çeviklik Degerlendirme (AgilityMod) ve Süreç Yetenek Degerlendirme (SPICE) Modelleri Saha Uygulamaları
(Agility Assessment AgilityMod and Process Capability Assessment SPICE Models Field Applications)
Damla Yasar, Serhat Caner, Hüseyin Ünlü, Ali Yıldız, Arif Kürsat Karabayır, Barıs Sokat, Burak Bilgin, Gizem Arık, Levent Anıl Özen, Osman Sönmeztürk, Seren Seyvan, Onur Demirörs
- Ar-Ge Merkezleri için ISO/IEC 15504 SPICE Temelli Süreç Modelinin ve Yönetim Yazılımının Geliştirilmesi
(Development of ISO / IEC 15504 SPICE Based Process Model and Management Software for R & D Centers)

Cihan Kabran Erdem Özgür Engin Hazar Nihal Bilal

Session 6A: Çevik Yazılım Geliştirme-1

- Alt Yüklenici Yönetiminde Çevik Bir Yaklaşım Deneyimi
  (Experience with Agile Approach in the Subcontractor Management)
  Gonca Hülya Selçuk, Hakime Koç
- Çevik Proje Takımlarında Tasarımsal Boyutuyla (De)Motivasyon Faktörleri
  (De)Motivation Factors in Agile Teams from the Design Perspective
  Necmettin Özkan

Session 6B: Deneysel Yazılım Mühendisliği

- ModelDOC: Entegre ve Katılımcı Elektronik Doküman Yönetim Sistemi
  (ModelDOC: An Integrated and Collaborative Electronic Document Management System)
  Osman Abul, Davut Gökhan Karatepe, Fadime Şeyda Demetoğlu, Naim Seyrek
- Yazılım Mühendisliği Profesyonellerinin Kişisel Öğrenmesi: Türkiye'den bir İnceleme
  (Personal Learning of Software Engineering Professionals: A Survey From Turkey)
  Selami Bagriyanik, Adem Karahoca
- Yazılım Uzmanları için Tasarlanmış Dijital Olmayan İş Birliği Oyunu
  (A Non-Digital Collaboration Game Designed for Software Professionals)
  Hazal Ipek Kahraman, Yasemin Bekturk, Irmak Angun, Yasemin Yigit Kuru, Murat Yilmaz

Session 6C: Yazılım Kestirimi

- ERP Projelerinde Efor Kestirimi
  (Effort Estimation in ERP Projects)
  Neslihan Küçükateş Ömüral, Onur Demirors
- Türkiye'deki Yazılım Geliştirme İşleri İçin İstenen Becerilerin Metin Madenciliği Kullanılarak Analiz Edilmesi
  (An Analysis of Desired Skills for Software Development Jobs in Turkey Using Text Mining)
  Görkem Giray
- Kaynak Kodlardaki Kötü Kokuların Analizinde Hata Komsuluk Etkisi
  (Bug Neighbourhood Effect on Analysis of Code Smells in Source Codes)
  Rahime Belen Sağlam, Özkan Kılıç, Yusuf Günaydın

Session 7A: Yazılım Geliştirme-1

- Mobil Uygulamalarda Gereksinim Belirlemede Başarıyı Arttıracak Bir Yöntem: Kitap Paylaşım Uygulaması Örneği
  (A Method to Increase Success in Determining Mobile Application Requirements: Book Sharing Application Example)
  Birsen Gülden Özdemir, Mustafa Zahid Gürbüz
- Yazılım Etmenleri Kullanarak Çok Sayıda Zindan Bölümünün Yöntemsel Olarak Üretilmesi
  (Procedural Generation of Multiple Dungeon Levels Using Software Agents)
  Kaya Oguz, Jane Lameski
- Blok Zinciri Kullanarak Tedarik Zinciri Yönetimi
  (Supply Chain Management Using Blockchain)
  Hazal Onay, İlayda Cansın Koç, Kuthan Temimhan, Hakan Yazıcı, Murat Erten

Session 7B: Lisansüstü Tezler Oturumu-1

- Yazılım Proje Yaşam Döngüsü Süreç Metrikleri ile Proje Performanslarının Ölçülmesine Yönelik Model Geliştirilmesi
  (Developing a Model for Measuring Project Performance with Software Project Life Cycle Process Metrics and Calculating Project Success Score)
  Özgür Gün, Pınar Yıldız Kumru

Session 7C: Yazılım Test Mühendisliği-4

- Test Olgunluk Modelinin TMMI Agile ve Test Süreçleri ile Bütünleştirilmesi
  (Test Maturity Model Integration (TMMI) Level Two Implementation with Agile and Software Test Process)
  Ahmet Unudulmaz, M. Özgür Cingiz Oya Kalipsiz
- Fourier Açılımı Yöntemiyle Mutant Seçimi
  (Mutant Selection Using Fourier Expansion)
  Savaş Takan, Tolga Ayav

Session 7D: Sağlık için Yazılım Mühendisliği-2

- Anti-Social: Bireylerin Sosyal Medya Bağımlılığını Azaltmak Amacıyla Düşünülmüş Bir Ciddi Oyun
(Anti-Social: A Serious Game for Reducing the Social Media Addiction of Individuals)
Ulaş Güleç, Murat Yılmaz, Can Koral Adalı, Veysi İşler
- Medikal Uygulama Geliştirme MUGE Platformu
(Medical Application Development (MADE) Platform)
Hande Sağlam, Deniz Katırcıoğlu Öztürk, Nihal Sarmaşık
- Sağlık Bilgi Sistemi ve Teknolojisi Olgunluk Faktörleri
(Maturity Factors of Health Information System and Technology)
Samet Can, Ayça Tarhan

Session 8A: Yazılım Mimarileri

- Mikroservis Mimari ile Geliştirilmiş bir Uygulamanın Mevcut Çoklu Uygulama Portföyüne Entegre Edilmesi
(Integration of an Application Developed on Microservice Architecture to the Existing Multiapplication Portfolio)
Ümit Kanoğlu, Ali İmre, Oumout Chouseinoglou
- Büyük Ölçekli Ve Kullanımda Olan Projelerde Ana Mimari Değişiklikleri
(Major Architectural Changes in Large Scale Projects)
Karani Kardaş
- Dagıtık Sistemlerin Izlenmesi Için Bir Yazılım Mimarisi Önerisi
(A Proposal of Software Architecture for Monitoring Distributed Systems)
İbrahim Onuralp Yiğit, Murat Demirsoy Burak Gökmen

Session 8B: Lisansüstü Tezler Oturumu-2

- Dağıtık Hizmet Reddi Saldırılarının Tekrarlayan Sinir Ağlarıyla Tespiti
(Detection of Distributed Denial of Service Attacks with Recurrent Neural Networks)
Serhat Caner, Murat Erten
- JSON Sözdizimli Gerçek Zamanlı İletişim Protokolü
(Real-Time Communication Protocol With JSON Syntax)
Serkan Ayaz, Semih Bilgen

Session 8C: Sponsor Sunumları

Session 8D: Sağlık için Yazılım Mühendisliği-3

- Sağlık Sistemleri için IoT ve Akıllı Sözleşme Kuralı Tabanlı Güvenli İletişim Tasarısı
(IoT - Smart Contract Rule Based Secure Communication Scheme For Healthcare System)
Aslı Kul, Eda Nur Azin, Ogulcan Ozdemir, Serap Sahin
- 3B Modelleme ve Düzenleme Araçlarının Estetik Cerrahi Alanında Kullanılması
(Usage of 3D Modelling and Morphing Tools in Plastic Surgery)
İlkin Sevgi İşler, Mehmet Harun Doğanay, Mürüvet Gökçen, Ulaş Güleç

Session 9A: Yazılım Analiz ve Ölçme

- Yazılım Hataları ile İlişkili Olmayan Olay Kayıtlarının Analizi ve Sınıflandırılması: Bir Vaka Çalışması
(Analysis and Classification of Issue Reports Not Related to Software Bugs: A Case Study)
Ethem Utku Aktas, Mehtap Ozaltin, Haluk Altunel, Cemal Yilmaz
- İki Güncel Dinamik Topluluk Takibi Yönteminin Karşılaştırmalı Bir Çalışması
(A Comparative Analysis of Two Most Recent Dynamic Community Tracking Methods)
Arzum Karatas, Serap Sahin
- Kullanıcı Merkezli Tasarımın Televizyon Etkileşimine Uygulanması
(Applying User-Centered Design to Television Interaction)
Evrim Akgündüz, Yasemin Topaloğlu

Session 9B: Yazılım Geliştirme-2

- Solidity Dili için Statik Analiz Aracı
(Static Analysis Tool for Solidity Language)
Ahmet GÜl, Alper Şen
- Web Sitesi Saldırılarına Karşı Hızlı Çözüm Üreten Bir Uygulama
(An Application for Quick Response to Website Attacks)
Murat Arslan, Burak Çarıkçı, Murat Erten
- Dijital Enerji IOT Uygulaması: KIO
(Digital Energy IOT Application: KIO)
Gökhan Akyol, Okan Akbay, Serkan Kaba



ÖNSÖZ

Ulusal Yazılım Mühendisliği Sempozyumu'nun (UYMS) on üçüncüsü, 23-25 Eylül 2018 tarihlerinde İzmir Yüksek Teknoloji Enstitüsü ev sahipliğinde düzenleniyor. Önceki yıllarda olduğu gibi bu yılki sempozyuma oldukça yoğun bir ilgi oldu. Ulusal Yazılım Mühendisliği Sempozyumu, Türkiye yazılım sektörü ile yazılım alanında çalışmalar yapan akademisyenleri bir araya getirerek yazılım mühendisliğindeki son gelişmelerin ve araştırmaların paylaşılmasını sağlayan bir platformdur. 2003 yılından bu yana gerçekleştirilen UYMS, araştırma sonuçlarının paylaşıldığı bir ortam sağlamasının ötesinde, ulusal yazılım mühendisliği araştırmalarının geleceğinin şekillenmesinde etkin rol oynamaktadır. Çalışmalarını sempozyumdaki çeşitli oturumlarda paylaşan tüm katılımcılara teşekkür eder, bu çabaların gelecek yıllarda yazılım mühendisliği alanında daha da başarılı çalışmalara yol açacağına olan inancımızı tekrarlamak isteriz.

UYMS temel alanla birlikte Yazılım Test Mühendisliği, Sağlık için Yazılım Mühendisliği, Yazılım Modelleme ve Lisans Üstü Tezler tematik alanlarında, bu yıl toplam 77 bildiri kabul edilmiştir. Bildiriler, bu yıl da, en az 3 hakem tarafından değerlendirilerek çok titiz bir inceleme sonucunda seçilmiştir. Hakem olarak görev alan tüm komite üyelerine gönülden teşekkür ederiz. Sunumları takiben bildiriler, CEUR sitesinde (http://ceur-ws.org) çevrimiçi olarak yayınlanacaktır.

Bu yılki davetli konuşmacımız Huawei firmasından Dr. Lina Cao'dur. Konuşmasında sürdürmekte olduğu yapay zekâ ve veri bilimi çalışmalarını ve bu çalışmaların sonucunda oluşacak paradigma değişikliklerini ele alacaktır.

Sempozyumdaki panel konusu "Akıllı Yazılım Çözümleri" olarak belirlenmiştir. Panelde birbirinden değerli araştırmacılar katkıda bulunacaklar ve bilgilerini bizlerle paylaşacaklardır. Tüm konuşmacılara katkıları ve paylaşımları için teşekkür ederiz.

Sempozyumun gerçekleşmesini mümkün kılan İzmir Yüksek Teknoloji Enstitüsü'ne ve etkinliğin gerçekleştirilmesi boyunca çeşitli görevleri yerine getiren Yönlendirme, Program ve Düzenleme komitelerinin üyelerine, web yöneticileri ile yayınlama, tanıtım ve mali işler sorumlularına teşekkür ederiz. Sponsorlarımız Huawei, Logo, IBTech, Panates firmalarına, ING Bankasına ve Teknopark İzmir'e verdikleri önemli destek için şükranlarımızı sunarız.

Saygılarımızla,
UYMS 2019 Program Komitesi



FOREWORD

The 13th National Software Engineering Symposium (UYMS) of Turkey is being held Izmir Institute of Technology on 23-25 September 2019. There has been a great interest in this year's symposium, as in previous years.

UYMS is a platform which helps bring together the software industry and the academicians working in this area. It is being organized since 2003 and it plays an important role in shaping the future of the software industry in Turkey. We would like to thank all the participants whose contributions led to the successful realization of this symposium. We would also like to express our belief that these contributions will lead to a better and more productive efforts in the field of software engineering.

Along with the main area of UYMS, in the thematic areas of Software Test Engineering, Software Engineering for Health, Software Modeling, and Graduate Theses, a total of 77 papers were accepted this year. At least three referees reviewed each paper and the papers were evaluated based on these reviews. We thank all the program committee members who served as referees. Following the presentations, the selected papers will be published online at CEUR (http://ceur-ws.org).

Our invited speaker of this year is Dr. Lina Cao of Huawei. She will share her work on Artificial Intelligence and the new ecosystem and business models which shall be generated as a result of the penetration of AI into our lives. We would like to thank her for sharing his valuable experiences with us which will make the impact of our symposium stronger.

The panel topic of the symposium is "Smart Software Solutions". In the panel, valuable researchers will contribute and share their knowledge with us. We would like to thank all speakers for their contributions.

We thank all the participants who shared their studies at various sessions in the symposium, and we would like to emphasize our belief that these efforts will lead to even more successful work in software engineering field in the years to come. We would like to thank Izmir Institute of Technology, for hosting the sympozium, and the members of the Steering, Program and Organization committees, the chairs of thematic areas, web administrators and publishing, promotion and financial affairs officers who fulfill various duties throughout the realization of the event. We extend our gratitude to our sponsors Huawei, Logo, IBTech, Panates, ING Bank and Teknopark İzmir for their invaluable support.

Your Sincerely,
UYMS 2019 Program Committee

13th Turkish National Software Engineering Symposium
Invited Talk:

Touching an Intelligent World

Lina Cao

Huawei, İstanbul, Turkey

Abstract. With current development of artificial intelligence (AI) and data science, "AI + Industry" will generate new ecosystems and new business models. Huawei aims to build a fully intelligent connected world. Huawei Turkey RD center, as the 2nd largest software RD center outside China, is taking a lot responsibilities in developing software for internal Huawei baseline products as well as local and global customers. During this presentation, I will introduce Huawei, Huawei Turkey RD center, software development in Huawei especially in AI applications. Several uses cases and on-going projects will also be discussed.

Biography. Lina Cao studied B.S. in Chemical Physics at University of Science and Technology of China. She received her Ph.D. in Optics from Columbia University in New York. She worked as a postdoctoral researcher in University of Chicago and Argonne National lab for two years. Dr. Cao had working experience as senior engineer in Seagate Technology, faculty member in Karadeniz Technical University and Sabanci University. Currently she is working as Technical SE and Project Manager for Technology Innovation Department at Huawei Technology. Her research experience includes professional software development in optical domain, Web development technologies, AI and big data-related algorithms. Dr. Cao has 3 US patents, several journal publications and more than two hundreds citations.

Yazılım Endüstrisi ve Akademi arasındaki Boşluğu Doldurmaya Yönelik bir Anket Çalışması: İlk Sonuçlar

Deniz AKDUR [0000-0001-8966-2649]

ASELSAN A.Ş., Ankara
denizakdur@aselsan.com.tr

Özet. Akademik geçmişleri sadece "Bilgisayar-tabanlı Disiplinleri" değil, aynı zamanda Elektrik ve Elektronik Mühendisliği gibi "Bilgisayar-tabanlı olmayan" disiplinleri de kapsayan birçok mühendislik mezunu, üniversitede öğrendikleri yeteneklerin yazılım endüstrisinin beklentileri ile tam uyuşmaması sebebiyle mesleki kariyerlerine başladıktan sonra bir çok zorlukla karşılaşabilmektedir. Şirketler, endüstri için "*hazır olmayan*" ve farklı bir çok yazılım mühendisliği rolünde çalışacak bu personeli eğitmek için çok önemli kaynaklar harcamakta; bu yüzden akademi, endüstride en çok kullanılan ve ihtiyaç duyulan yetenekleri bilip buna uygun bir müfredat belirlemek zorundadır (hem kavramlar ve bilgi alanları gibi teknik yetenekleri, *ing. hard skills* hem de iş yerinde çokça ihtiyaç duyulan beşeri yetenekleri *ing. soft skills*). Bir diğer taraftan, "*yazılımın geliştirilmesi, işletilmesi ve bakımı için sistematik, disiplinli ve ölçülebilir bir yaklaşım*" olarak tanımlanan ve hem akademinin hem de yazılım endüstrisinin ortak çıktısı olan disiplinler arası bir uygulama olarak tanımlanan yazılım mühendisliğini şekillendirip kullanan bu akademisyen ve pratisyenlerin amaçları, katkıları ve kaygıları çok farklı olabilmektedir. Tüm bunlar düşünüldüğünde, hem yazılım endüstrisinin yazılım mühendisliği ile ilgili olabilecek akademik programlardaki müfredatların ihtiyaçlara göre belirlenebilmesi; hem de endüstri profesyonellerinin akademik dünyaya bakış açısını yansıtarak daha fazla endüstri-akademi işbirliği (EAİ) sağlanabilmesi için bir anket çalışması hazırlandı. Değişik yazılım mühendisliği rollerindeki (örneğin, yazılım geliştiricisinden, sistem ve test mühendisine kadar) farklı sektör ve uygulama alanlarından, 14 ülkeden 637 katılımcıya ulaşan bu anketle, endüstrinin akademiden beklentileri ve ihtiyaçları adreslenerek "*Yazılım endüstrisi akademiden ne istiyor?*" sorusu cevaplanmaya çalışılmıştır. Bu bildiride anket sonrası ön analiz sonuçları sunularak, hem akademi hem de endüstride farkındalık yaratılıp, müfredatlarnın şekillenmesi, güncellenmesi ve de "*akademiden daha hazır*" gelebilecek profesyoneller sayesinde şirketlerin eğitim, oryantasyon gibi maliyetlerinin azaltılması hedeflenmektedir.

Anahtar Kelimeler: yazılım endüstrisi, yazılım mühendisliği eğitimi, bilgisayar-tabanlı disiplinler, müfredat, anket, endüstri akademi işbirlikleri (EAİ)

A Survey on Bridging the Gap between Software Industry and Academia: Preliminary Results

Deniz AKDUR [0000-0001-8966-2649]

ASELSAN A.Ş., Ankara
denizakdur@aselsan.com.tr

Abstract. Many engineering graduates, whose academic backgrounds are not only based on "Computing Disciplines" but also non-computing ones, face difficulties after beginning their professional careers in the software industry due to misalignment of the skills learnt in the

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

university with what is required in the workplace. Companies spend crucial resources to train these personnel, who are not "ready" for different software engineering (SE) roles. Academia must know what skills (e.g., hard skills such as key SE topics and knowledge areas (KAs) besides soft skills) are mostly used to adapt the educational programs via an effective curriculum. Although SE is shaped and used by both practitioners and academicians, these two separate worlds have different goals and concerns. In order to increase Industry-Academia Collaboration (IAC) in SE, it is necessary to understand different perceptions of practitioners about academia. To achieve these objectives, an online survey, which explores the gap between the software industry expectations and academic activities, was designed and conducted. The survey was responded by 637 software practitioners from 14 countries working in different SE roles with different highest academic degrees, work experiences, application types and industrial sectors. In this article, the preliminary analysis on survey results are presented.

Keywords: software industry, software engineering education, computing discipline, non-computing, curriculum, practitioner survey, industry academia collaborations (IAC)

1 Introduction

Software practitioners in the industry face difficulties to align the skills learnt in the university with what is required in the workplace. Since Software Engineering (SE) is defined as "*the application of the systematic, disciplined and quantifiable approach to the development, operation and maintenance of software*" [1], this definition affects how or to whom related topics are taught in the university; hence most resesearch focuses on improving the curriculum for "Computing Disciplines" (e.g., Computer Science (CS), Computer Engineering (CENG), Information Technology (IT) and Information Systems (IS)). However, the practitioners have a variety of different SE roles (e.g., from software developers to systems engineers and testers), whose academic backgrounds are not only based on these computing disciplines but also non-computing ones such as Electrical and Electronics Engineering Engineering (EEE). It is reported that practitioners in the embedded software industry, who were graduated from any non-computing disciplines are lacking knowledge in key SE topics and knowledge areas (KAs), which they learn or improve themselves during the job (e.g., after university education) [2, 3]. In our previous survey, which investigated software modeling practices in embedded sectors (e.g., consumer electronics, defense & aerospace), ~40% of respondents were graduated from non-computing disciplines, which shows that there is a huge number such practitioners [4]. Therefore, the need for an effective curriculum for different SE roles should not focus on only computing disciplines but also a wider perspective. It is crucial to analyze the gap between the software industry expectations and the academic curriculum for two reasons: (1) for software industry, it is important to hire properly trained (e.g., "ready") graduates, which allows them to spend less time while incorporating these personnel more efficiently into the workplace; (2) for academia, understanding the necessary skill set is critical for curriculum maintenance and development.

SE, which is the application of engineering to software [1], is shaped and used by both academicians and practitioners. However, these two different worlds have totally different goals, contributions and concerns. Since academicians and practitioners have different mindsets (e.g., motivation, focus, etc.), the level of Industry-Academia Collaboration (IAC) in SE is low and they are not collaborating with each other to solve industrial problems. Depending on demographic factors such as educational-skill set, SE role or previous poor experience on IACs, there are mismatches in perceptions of various practitioners. It is very necessary to investigate different perceptions of practitioners about academicians in order to increase IACs by improving mutual understanding of each side.

In order to address the need to align different SE roles' education with software industry and to understand different opinions of the practitioners about the academic world, a practitioner survey was designed and conducted. 637 software practitioners from 14 different countries participated in this survey, which closes the gap between industry and academia. The participants have a variety of different SE roles from software developers to systems engineers and from testers to high level managers, whose academic backgrounds are based on both computing disciplines (CS, CENG, IS, etc.) and non-computing ones (especially EEE from embedded software industry). In this article, the preliminary analysis on survey results (mainly demographics data) are presented besides inter-

esting findings, which will be planned to analyze various cross-factor analysis for different demographics to increase IACs in SE.

The remainder of this paper is structured as follows. Section 2 gives the related studies. Section 3 presents the research methodology. Section 4, which is the main goal of this article, presents the preliminary results. Finally, Section 5 concludes this study by giving future directions.

2 Related Studies

In the literature, the knowledge gap between the industry needs and educational programmes was highlighted by several studies (e.g., [5-10])). When analysing necessary SE skills, there are two primary concepts: "*hard skills*" and "*soft skills*". Hard skills can be seen as "technical" skills that are gained in the academic curriculum. On the other hand, although soft skills can also be improved in these curricula, there is almost no dedicated course for these topics. Rather, students are expected to get these skills through in-class activities or during team projects. Many studies focused on these skills and their results show that both hard and soft skills are needed in order to understand the needs of the software industry.

This survey study builds on previous studies with significant extensions: it is not limited to either the educational background (e.g., computing disciplines), nor a subset of SE roles (e.g., not only developers or IT personnel), nor just a specific region (e.g., not only USA, UK, Finland, New Zealand), nor having too general results to address specific needs (e.g., focuses on application type such as embedded vs desktop or industrial sector (e.g., Consumer Electronics, Defense & Aerospace, IT & Telecommunications)). In this perspective, this survey, by including practitioners' opinions about academia, closes the gap between industry and academia with a wider coverage.

3 Research Methodology

In this study, the online survey method was chosen to obtain information from various practitioners in a quick manner to analyze these data easily [11].

The main goal of this survey is to bridge the gap between software industry expectations and academic activities. This main goal is decomposed into three sub-goals: (1) Identifying the usage and importance of SE KAs and topics by measuring knowledge gaps with the industry needs and academic curriculum (e.g. "hard skills" analysis); (2) Understanding the most important soft skills; (3) Analysing the opinions of practitioners about IAC on both educational and research activity sides. Based on these goals, by creating corresponding survey sections, the following research questions (RQs) are raised:

RQ1*: What are the most used KAs and SE topics in the software industry? What are the knowledge gaps and coverage of the industry expectations after university education?*
RQ2*: How does educational skill set of the practitioner affect software modeling approach and practices?*
RQ3: *What are the most important soft skills in the industry?*
RQ4: *What are the opinions of software practitioners for more IAC as a part of the education?*
RQ5: *How do practitioners see academicians? What are their perceptions about academicians and academic outputs?*

In order to develop a survey that would adequately cover the goal of this study, the organization of survey was carefully designed by considering survey guidelines [11] and also previous experience in designing & executing industrial surveys (e.g., [4]). For this study, due to space constraints, please refer to [12] for the details of all design phases activities of the survey, which took over six months (e.g., the selection of critical SE topics and identifying the practitioners' opinions about how they see academics, expert opinion, survey piloting, etc.).

The organization of the survey is depicted in **Fig. 1**. After gathering the demographics of the participants, the survey questionnaire consisted of five main sections [13], whose preliminary results are presented in Section 4. To design and execute the survey, the Google Forms tool was used. The ethics approval for the survey was issued by the Human Subjects Ethics Committee of Middle East Technical University (METU) in March 2019. The survey was then executed in the period of March-May 2019.

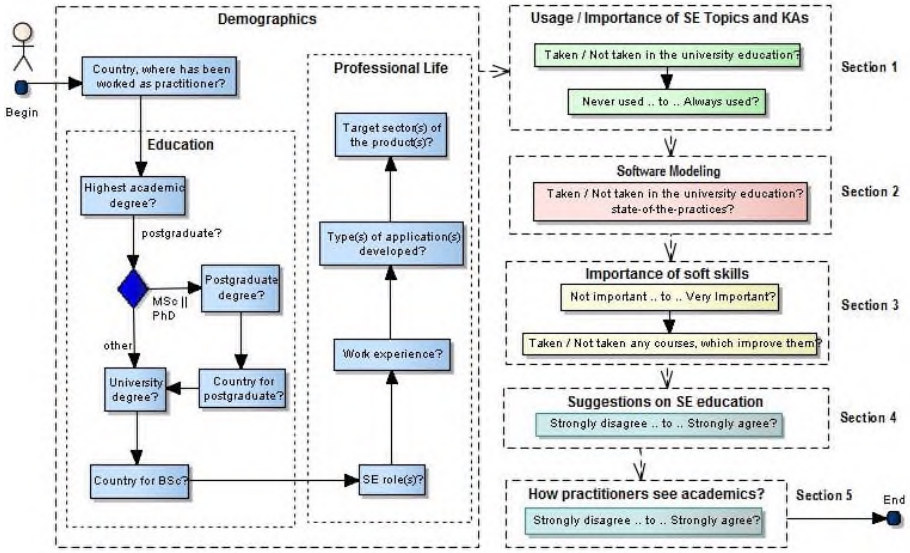


Fig. 1. The organization of the survey

4 Preliminary Analysis

Opinions of 637 software practitioners with different demographics from 14 different countries are included in the survey. In this section, due to space constraints, the demographics of participants and the –limited- preliminary results of RQ1 (without cross-factor analysis) are presented. Please refer to [14] for the other RQs' raw data results.

4.1 Demographic of participants and their companies

It is necessary to get a detailed demographics in such a study since practitioners' and projects' characteristics directly affect the results. The first question asked respondent about the country that the participant is working. The final dataset had respondents from 14 different countries as depicted in **Fig. 2**. Note that due to researcher's location (i.e., Turkey), the ratio of Turkey (56%) is higher than others.

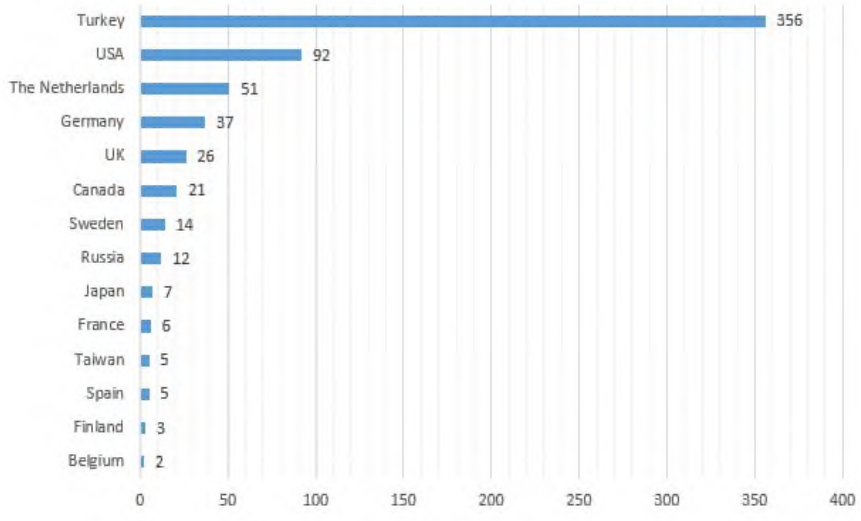


Fig. 2. The countries where the participant is working

Participants were asked to provide their highest academic degrees. The result reveals that ~57% of participants have a postgraduate degree (MSc 38.6% and PhD 18.4%) and 43% of respondents have a BSc degree as shown in **Fig. 3**.

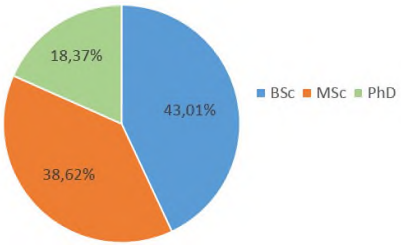


Fig. 3. Highest academic degrees

In order to understand the respondents' educational skill-set, participants were then asked to provide their university degrees. The participants, who answered the highest academic degree question as "BSc", respond only the question for BSc university degree; however participants, who have any postgraduate degree (e.g., either MSc and/or PhD) present their both university degrees (i.e., BSc and postgraduate degrees respectively). **Fig. 4** depicts BSc degrees of participants, which shows that EEE graduates, which is one of the non-computing disciplines, have a significant percentage (24%) in the participant pool.

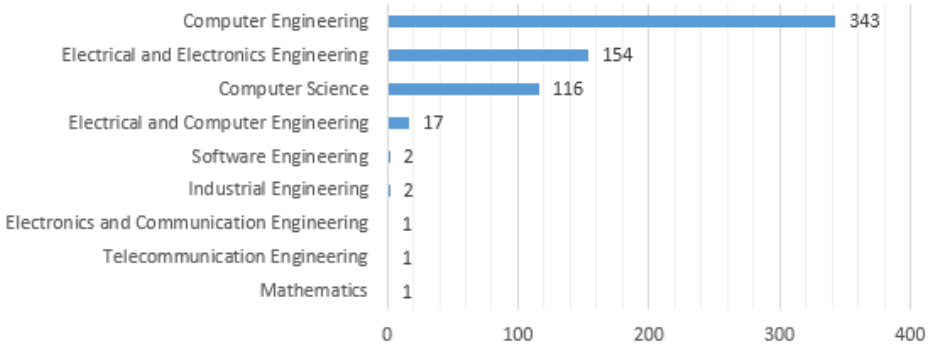


Fig. 4. BSc degrees

The need for an effective curriculum for different SE roles should not focus on only computing disciplines but also a wider perspective; hence it is better to categorize the university degrees according to the "computing discipline" criteria by taking reference on **Table 1** information.

Table 1. Computing disciplines vs Non-computing disciplines for university degree

Computing Disciplines	**Non-computing Disciplines**
Computer Engineering (CENG)	Electrical and Electronics Engineering (EEE)
Computer Science (CS)	Telecommunication Engineering (TE)
Information Systems (IS)	Electronics and Communication Engineering (ECommE)
Software Engineering (SE)	Industrial Engineering (IE)
Electrical and Computer Engineering (ECE)	Mathematics (Math)
Information Technology (IT)	Systems Engineering (SysE)

Accordingly, when this new "derived" criteria is used, the results showed that 25% of participants (i.e., 159 participants) took non-computing disciplines curriculum for BSc although they are working in the software industry (**Fig. 5**).

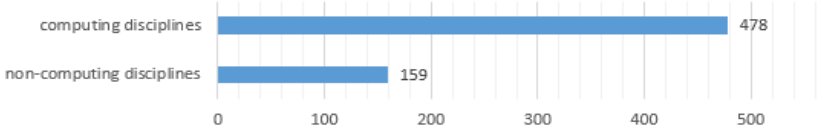


Fig. 5. BSc degrees based on computing vs non-computing disciplines

As given in **Fig. 3**, ~57 % of participants (i.e., 363 software practitioners) have a postgraduate degree and whenever this type of university degrees are analysed, **Fig. 6** is depicted.

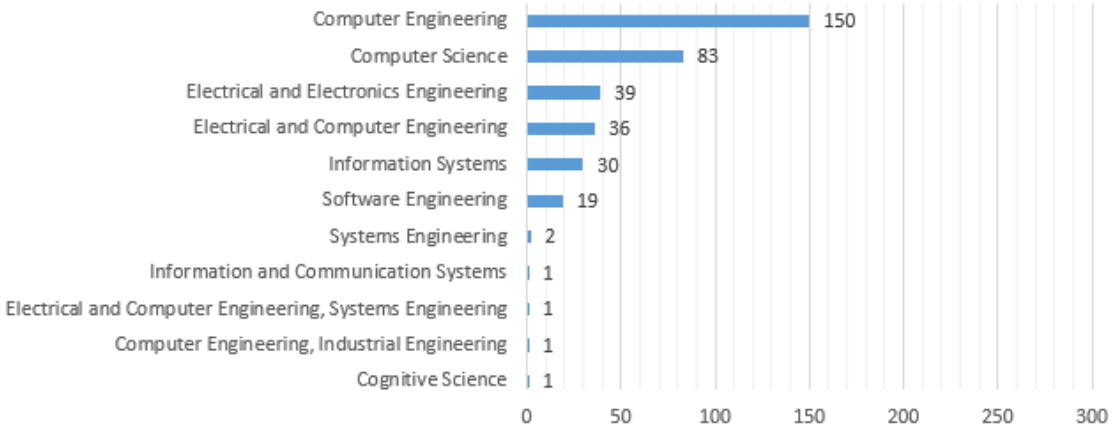


Fig. 6. Postgraduate (MSc and/or PhD) degrees

In the final pool of participants (e.g., after BSc and postgraduate degree (if any)), when the percentage of computing vs non-computing disciplines is analysed, it is seen that the percentage of the participants, who took any computing disciplines curriculum increased about 6,3% (i.e., 518 participants, 81.3%) as depicted in **Fig. 7**. This might mean that these non-computing disciplines took any computing discipline postgraduate degree to improve themselves in the industry.

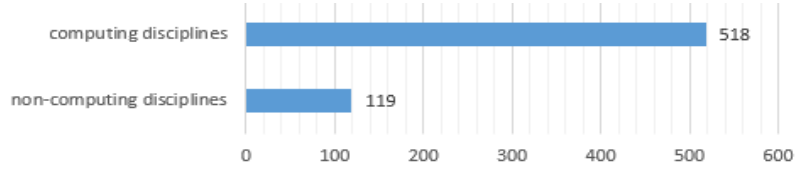


Fig. 7. Computing vs non-computing disciplines in the final participant pool

SE practices used in the industry and opinions of practitioners about IAC do not only depend on their university education (e.g., not only based on the courses taken in BSc or postgraduate) but also depend on the experiences and training undertaken during the job. In other words, we cannot isolate the gained skill set and perceptions of the practitioner from the workplace, which might also affect the results of such a study. Therefore, besides university degree(s) (e.g., computing vs non-computing), it is also very necessary to know the relations between the countries, where the practitioner has worked and the country, where this practitioner completed his university education (e.g., BSc and postgraduate, if any). Note that a practitioner, who completed his/her BSc degree in one country, might go abroad for postgraduate education or for work during their professional career; hence knowing the –possible- differences between these countries is crucial to compare the results. Therefore, after getting university degree for BSc and postgraduate (if any), the country/countries, where the practitioner completed their university education are asked as another questions. The goal of these questions is to create a map, which shows the educational and career path of the software practitioner after BSc education. **Fig. 8** shows such information, where you can trace the path of the participants, who go abroad for any postgraduate education or professional career or stay the country, where she/he completed her/his BSc education.

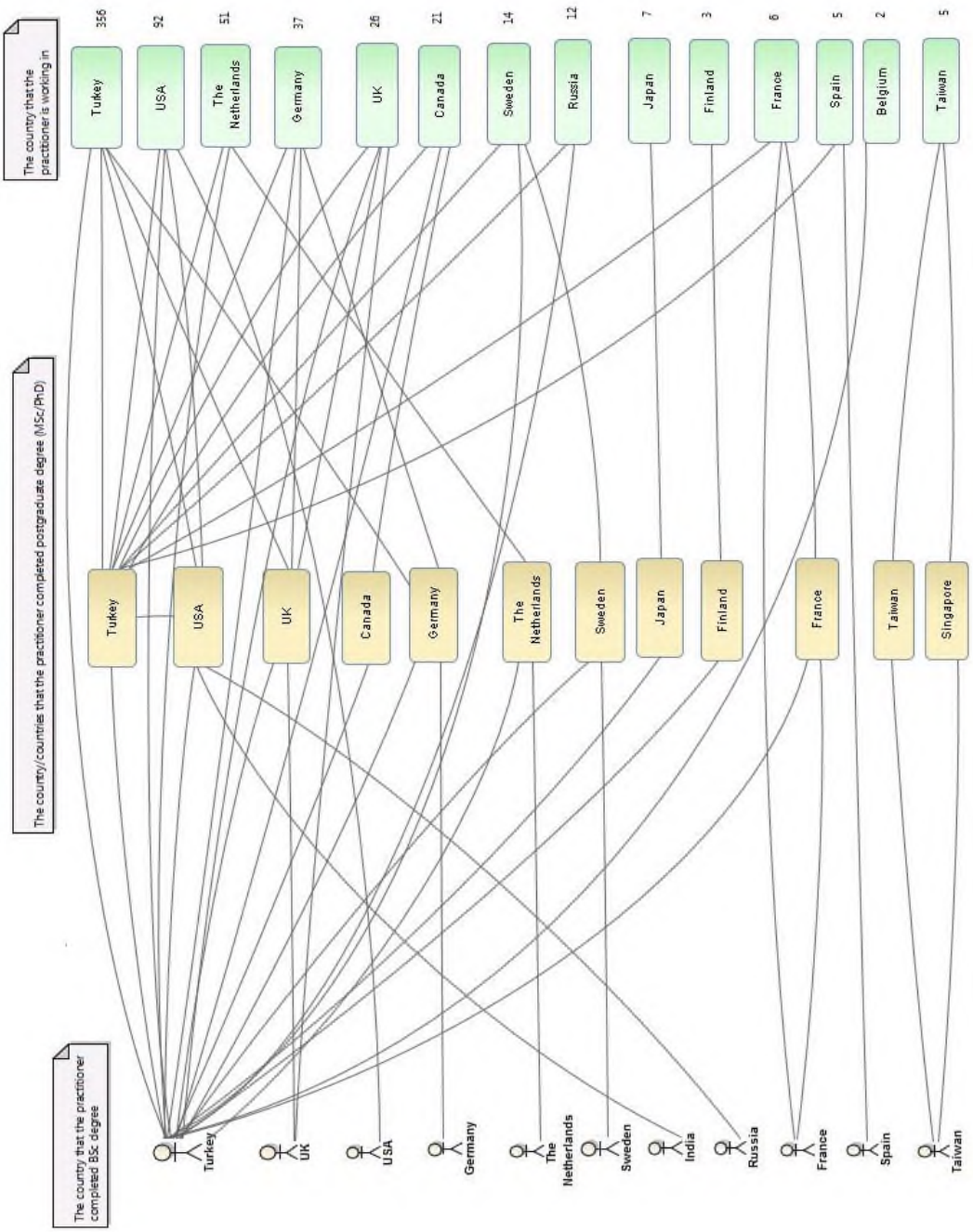


Fig. 8. The path of the participants after university education
(The country/countries, where the participants completed university degree(s) and where, they are currently working in)

Apart from these information, knowing SE role(s) in their careers, the work experience (e.g., for the perceptions of experienced professionals vs fresh graduates), the type of applications (e.g., embedded vs desktop vs web) and industrial sector (e.g., consumer electronics, defense & aerospace, automotive, etc.) are also important factors to better understand and align the gap between software industry and academia. To achieve this goal, participants were asked about these demographics data in multiple response questions.

Most of the participants have "Software Developer/Programmer" role in their career (91.2 %) as depicted in **Fig. 9**. "Software Architect" is the other majority role, which is mostly achieved after being "Software Developer/Programmer" for a while (e.g., after ~5-10 years of work experience). In other words, since this question is multiple response question, multiple roles could be recorded (e.g., a person can be a software developer/programmer at the beginning of its professional career and then become a software architect). Similarly, a software practitioner, who is currently working as a "Middle / High Level Manager", might be a software developer at the very

beginning of her/his career, then become a software architect, and perhaps after 15+ years of work experience, become a software manager. Individual responses and the results of this question also help to understand possible career path for the software practitioner [14].

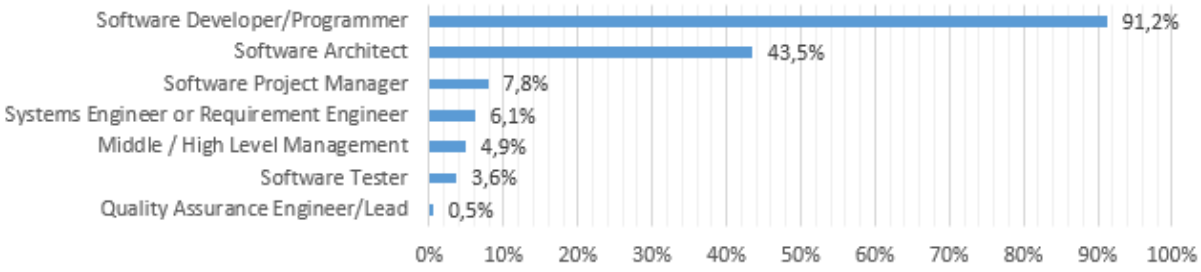


Fig. 9. SE roles in the professional career

According to the results, the majority (43%) had 10-15 years of work experience, followed by 5-9 years (32.8%) and 15+ years (17.7%) of work experience as depicted in **Fig. 10**.

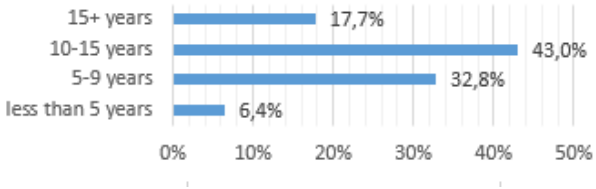


Fig. 10. Work experience

Participants were also asked about the type of the applications developed as a multiple-response question. “Embedded applications” is the majority (63 %), which is followed by “Desktop applications” and “Web applications” (**Fig. 11**). Some participants (i.e., ~1 %) used “free text” area for this question as “Other”. After analyzing these responses, some of them are merged into the existing categories (e.g., “mobile” is counted as “embedded”; or “server” is counted as “web”).

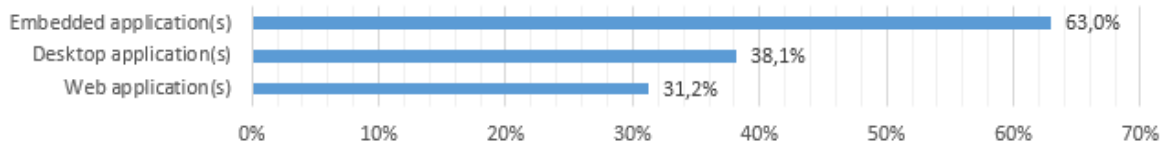


Fig. 11. Application type

When the target sectors of the products developed by the company employing the participants was asked, the results are depicted in **Fig. 12**. Three most popular industrial sectors are “Defense & Aerospace”, followed by “Consumer Electronics”, and “IT & Telecommunications”.

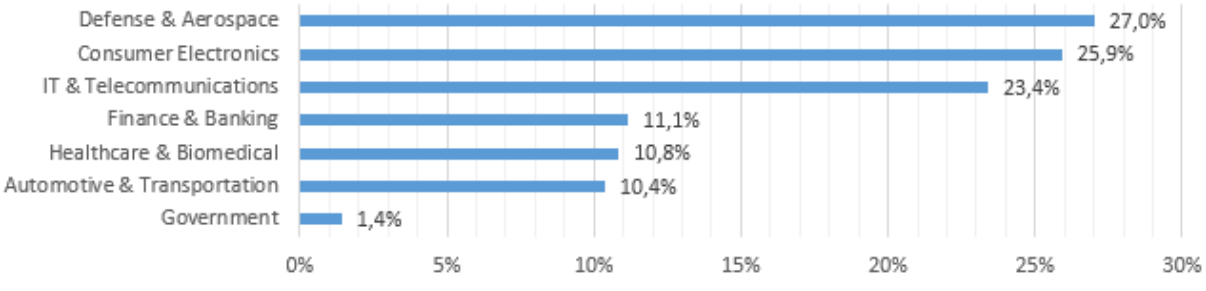


Fig. 12. Industrial sector

4.2 Preliminary Results

In this section, due to space constraints, RQ1 of this study, which investigates the most used KAs and SE topics by exploring the knowledge gaps in the software industry is addressed as preliminary analysis without any cross-factor.

In order to align different SE roles' education (from software developer to systems and test engineers) with industry expectations, Section 1 of the survey aims at identifying the most important hard skills in the industry by shedding light on usage/importance and knowledge gaps of different SE topics. After identifying the necessary KAs and SE topics based on curriculum guidelines such as SWEBOK and ACM Computing Curricula ([1, 15]) and by reviewing similar surveys (See Section II) and the improvement with personal industrial experience, Section 1 mainly asks whether the participant took a specific course related to this item (e.g., either a KA or a SE topic). Then, by asking the usage of this item (i.e., as Likert scale from Never(0) to Always(5)), it is aimed to analyze the knowledge gaps to address RQ1.

As depicted in **Fig. 13**, as expected almost every participants (98,5%) took a course related to "Software Construction", which is directly related with programming and coding. 63,1% of participants took a specific course related to "Software/Systems Design and Architecture", followed by "Software Quality" (45%) and "Software Engineering Management" (42%).

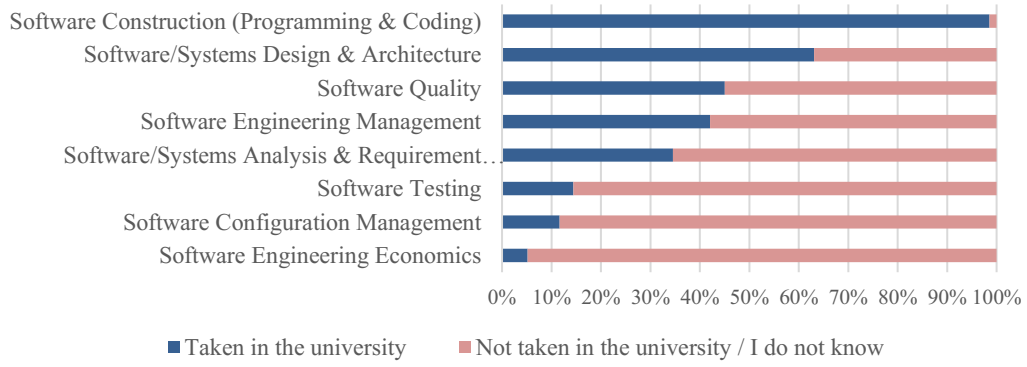


Fig. 13. KAs taken / not taken in the university education

According to the results, the most used KA – as expected again - is "Software Construction", as seen from **Fig. 14**. However, there are some interesting findings, when analyzing the knowledge gaps in the industry. Although some KA related courses are not taken in the university education; their usage (hence the importance level) are very high, which causes knowledge gaps in the software industry. According to the results, "Software Configuration Management", which was not taken any related course by 88,4% of participants, almost always (e.g., 4,416 level of 5) is used in the industry. This significant gap is the greatest gap among the other KAs.

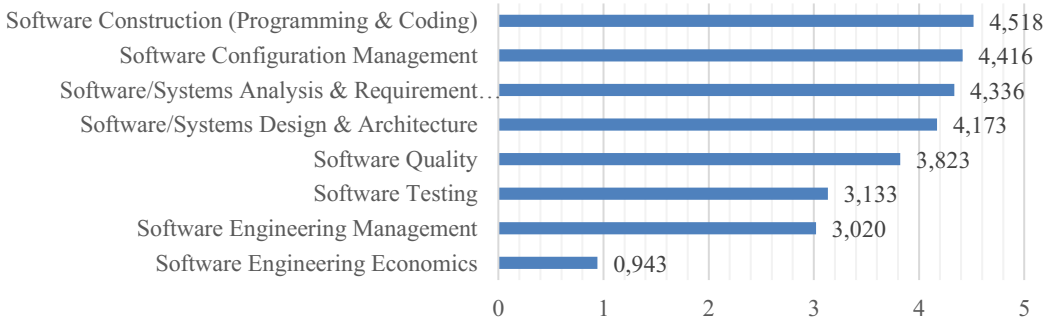


Fig. 14. Usage/Importance of KAs

Similarly, "Software/Systems Analysis & Requirement Engineering", which 65,5% of participants did not take a course dedicated to this KA, is mosty used (e.g., 4,336 level of 5) in the indus-

try as depicted in **Fig. 14**. "Software Testing" is also another knowledge gap, which the academia should give more importance since the given usage/importance are not correlated with what the university curriculum include. On the other hand, "Software Engineering Economics", which was not taken as a course by many participants (94,8%) is the least used KA, which means that importance of this KA is less than the others. In this case, since its usage is also low (i.e., 0,943 level of 5) as seen from **Fig. 14**, there is no significant knowledge gap for this KA.

After understanding the knowledge gaps for specific KAs, the participants were asked about different SE topics in a similar manner. The most popular courses (e.g., the curriculum include these courses so that most participants took them) are "Data Structures" (90,6%), "Operating Systems" (89,8%) and "Hardware concepts & Computer Architecture" (88,1%). On the other hand, only 1,4% of participant took a course related to "Blockchain fundamentals" as seen **Fig. 15**.

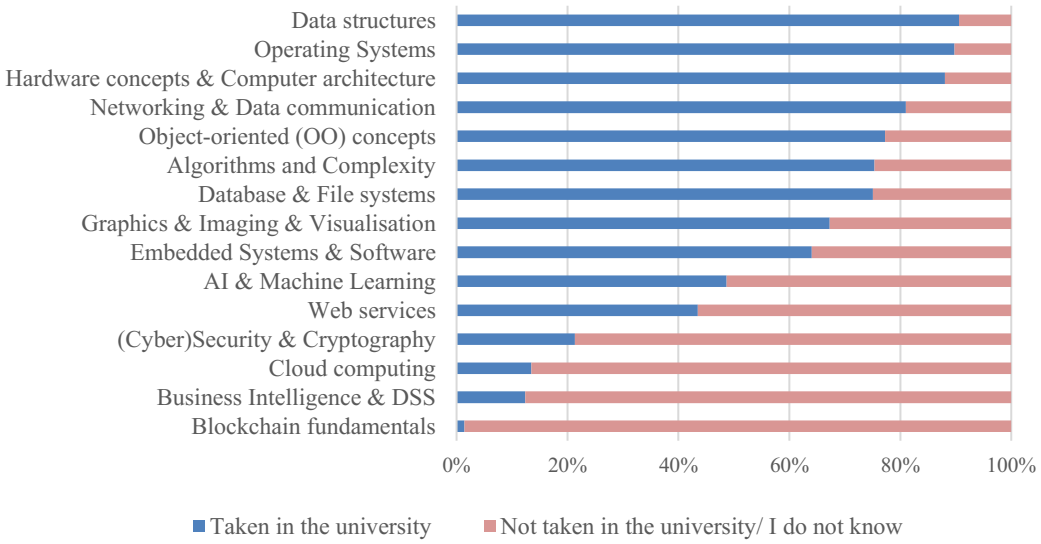


Fig. 15. SE topics taken / not taken in the university education

After analyzing the usage of these SE topics, **Fig. 16** is depicted. Accordingly, "Data structures", "Object-oriented (OO) concepts" and "Database & File systems" are the top three used SE topics in the software industry.It is seen that there is no big gap for a specific SE topics as opposed to knowledge gaps encountered in KAs. In other words, the usage/importance of SE topics are correlated with the ratio for the courses taken in the university.

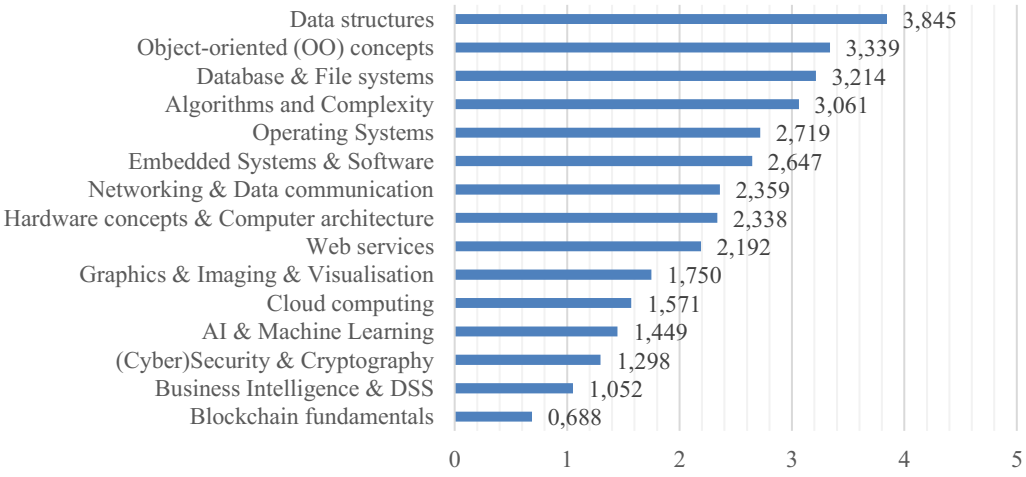


Fig. 16. Usage/Importance of SE topics

Mathematical Foundations helps software engineers to comprehend various SE related topics, which in turn is translated into programming language code or logic rather than on challenging arithmetic abilities. Note that instead of asking this KA in the previous section, which addresses various KAs such as Software Construction and Software Quality (**Fig. 13** and **Fig. 14**), we wanted to understand which specific mathematic courses are used hence important for different SE roles in which application types. Therefore, in a specific sub-section, the participants were asked about the usage/importance of five mathematical-related courses after asking their existence in their curriculum. As seen **Fig. 17**, all participants (100%) took "Calculus" and "Probability & Statistics" in the university; whereas 94% of participants took "Differential Equations", 86,5% of participants took "Propositional/Predicate Logic", and 72,6% of participants took "Discrete & Combinatorial Mathematics". On the other hand, the most used mathematical-related course is "Probability & Statistics" (i.e., 1,876 level of 5), whereas "Differential Equations" is the least used topic (i.e., 0,397 level of 5). As seen, there is a big gap (e.g., negative gap) in "Differential Equations" since this course is almost taken by everyone (i.e., 94%) although its usage is very low in the software industry. As a big picture, no mathematical related courses' usage is higher than 2 level of 5, which might mean that there is a misalignment of what is needed in the industry and given in the university. Mathematics should be included in curricula, but perhaps the weight of some mathematics-related courses (e.g, the given course hours) might be updated.

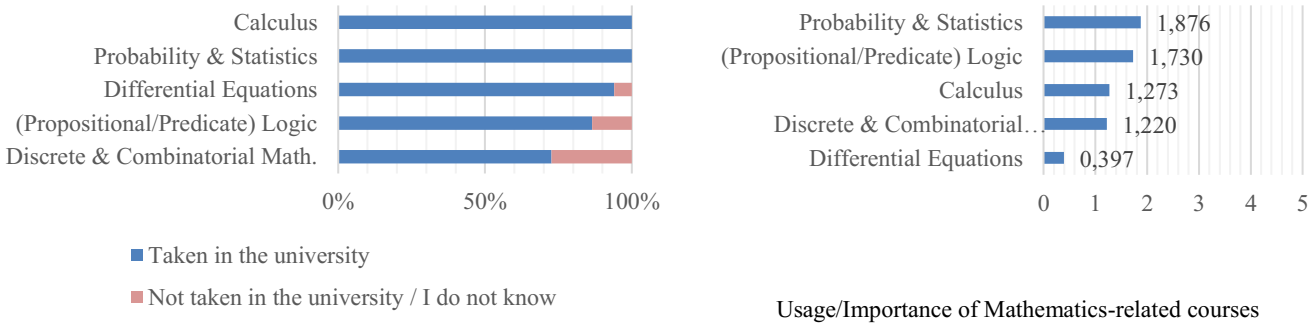


Fig. 17. Mathematical Foundations KA

Note that all the preliminary results given in this section are raw data taken from the survey without filtering any participants' demographics. However all the usage/importance of these KAs and SE topics will be more helpful when cross-factor analysis (e.g., the most important skill set according to application type(s) such as embedded vs web; or industrial sector such as defense or consumer electronics) , which has been already started for different demographics are identified.

5 Conclusion and Future Directions

In this article, due to space constraints, the preliminary results of a survey, which aims to bridge the gap between the software industry expectations and academic activities, was presented.

With the help of the collected data, whose cross-factor analysis for different demographics (both educational and professional life) have been already started, academia can adapt their programs with an effective curriculum for different SE roles (e.g., from software developers to testers and systems engineers) in different sectors so that various graduates will be more efficiently incorporated into the software industry. Moreover, revealing practitioners' perceptions about academia and academic activities with demographic differences (e.g., regional, application type, sector, SE roles, etc.) will, hopefully, increase IACs in SE.

Acknowledgments. The author would like to thank all software practitioners, who contributed to this survey.

References

[1] P. Bourque and R. E. Fairley, *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*: IEEE Computer Society, 2014.

[2] D. Akdur, V. Garousi, and O. Demirörs, "Cross-factor analysis of software modeling practices versus practitioner demographics in the embedded software industry," in *6th Mediterranean Conference on Embedded Computing (MECO)*, Montenegro, 2017.

[3] D. Akdur, O. Demirörs, and B. Say, "Towards Modeling Patterns for Embedded Software Industry: Feedback from the Field," in *44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Prag, Czech Republic, 2018.

[4] D. Akdur, V. Garousi, and O. Demirörs, "A survey on modeling and model-driven engineering practices in the embedded software industry," *Journal of Systems Architecture* vol. 91, pp. 62-82, 2018.

[5] T. C. Lethbridge, "What Knowledge Is Important to a Software Professional?," *Computer,* vol. 33, pp. 44-50, 2000.

[6] B. Kitchenham, D. Budgen, P. Brereton, and P. Woodall, "An investigation of software engineering curricula," *Journal of Systems and Software,* vol. 74, pp. 325-335, 2005.

[7] S. Surakka, "What subjects and skills are important for software developers?," *Commun. ACM,* vol. 50, pp. 73-78, 2007.

[8] A. M. Moreno, M.-I. Sanchez-Segura, F. Medina-Dominguez, and L. Carvajal, "Balancing software engineering education and industrial needs," *J. Syst. Softw.,* vol. 85, pp. 1607-1620, 2012.

[9] C. L. Aasheim, S. Williams, and E. S. Butler, "Knowledge and Skill Requirements for it Graduates," *Journal of Computer Information Systems,* vol. 49, pp. 48-53, 2009.

[10] A. Radermacher, G. Walia, and D. Knudson, "Investigating the skill gap between graduating students and industry expectations," presented at the 36th International Conference on Software Engineering, India, 2014.

[11] J. Linaker, S. M. Sulaman, R. Maiani de Mello, M. Höst, and P. Runeson, "Guidelines for Conducting Surveys in Software Engineering," 2015.

[12] D. Akdur, "The Design of a Survey on Bridging the Gap between Software Industry Expectations and Academia," in *8th Mediterranean Conference on Embedded Computing (MECO)*, Montenegro, 2019.

[13] D. Akdur, "Survey Form: Bridging the Gap between Software Industry Expectations and Academic Activities," *https://drive.google.com/file/d/1TvYUHScDojZPC6nLSWGpewKUbS0ewgrP/view,* 2019, Last accessed: Mar. 15, 2019.

[14] D. Akdur, "Raw Data: Survey on What Software Industry Wants From Academia," *https://www.researchgate.net/publication/333892917_RawDataResults4SurveyOnWhatSoftwareIndustryWantsFromAcademia,* 2019, Last accessed: June 10, 2019.

[15] ACM, AIS, and IEEE-CS, "Computing Curricula," 2005.

Türkiye'de Yazılım Geliştirme Takımlarının Bulut Kullanımı, Kullanım İsteği ve Uygunluk Algısı

Erhan Pişirir[1][0000-0002-2365-4643], Cüneyt Sevgi[2],
Oumout Chouseinoglou[3][0000-0002-8513-351X] ve Erkan Uçar[2][0000-0002-9042-3407]

[1] Hacettepe Üniversitesi, İstatistik Bölümü, Ankara, Türkiye
[2] Bilkent Üniversitesi, Bilgisayar Teknolojileri ve Bilişim Sistemleri Bölümü, Ankara, Türkiye
[3] Hacettepe Üniversitesi, Endüstri Mühendisliği Bölümü, Ankara, Türkiye
erhanpisirir@hacettepe.edu.tr, csevgi@bilkent.edu.tr,
uhus@hacettepe.edu.tr, eucar@bilkent.edu.tr

Özet. Yeni teknolojik gelişmelerin kullanıcılar tarafından benimsenip mevcut alternatiflerin yerine kullanılması sürecini etkileyen içsel ve dışsal faktörler bulunmaktadır. Yeni bir teknolojinin kullanıma uygunluğu, o teknolojiyi kullanma isteği ve o teknolojinin hâlihazırda kullanılıyor olması farklılıklar gösterebilmektedir. Bu çalışmada, Türkiye'de yazılım geliştirme projelerinde bulut bilişim servislerinin kullanım durumu incelenmiştir. Yapılan anket çalışması sonuçları istatistiksel yöntemlerle karşılaştırılmış ve yazılım projelerindeki aktivitelerde bulut teknolojilerinin mevcut kullanımı, yazılım geliştiricilerin bulutu işlerine uygun bulma derecesi ve yazılım geliştiricilerin bulut kullanım isteği arasındaki ilişkiler çıkarılmıştır.

Anahtar Kelimeler: Bulut Bilişim, Yazılım Geliştirme Aktiviteleri, Bulut Tabanlı Yazılım Geliştirme, Teknoloji Kullanım Modeli.

Actual Use, Use Intention and Perceived Suitability of Cloud by Turkish Software Development Teams

Erhan Pişirir[1][0000-0002-2365-4643], Cüneyt Sevgi[2],
Oumout Chouseinoglou[3][0000-0002-8513-351X] and Erkan Uçar[2][0000-0002-9042-3407]

[1] Hacettepe University, Department of Statistics, Ankara, Turkey
[2] Bilkent University, Computer Technologies and Information Systems, Ankara, Turkey
[3] Hacettepe University, Department of Industrial Engineering, Ankara, Turkey
erhanpisirir@hacettepe.edu.tr, csevgi@bilkent.edu.tr,
uhus@hacettepe.edu.tr, eucar@bilkent.edu.tr

Abstract. There are several internal and external factors that affect the adoption and use of a new technology instead of the existing alternatives. The suitability

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

of a new technology, the intention to use it and actually using it may show differences. In this study, the state of cloud use in software development projects in Turkey is investigated. To this end, a questionnaire is administered and the results are analyzed statistically to derive the relationships between the actual use, use intention and perceived suitability of cloud services by software developers regarding development activities in software projects.

Keywords: Cloud Computing, Software Development Activities, Cloud Based Software Development, Technology Use Model.

1 Giriş

Yazılım geliştirme projeleri içerisinde birçok aktivite gerçekleştirilmekte olup, takip edilen yazılım geliştirme süreci bu aktivitelerin hangi sırada ve ne şekilde ilerleyeceğini şekillendirmektedir. Temel yazılım geliştirme aktiviteleri, gereksinim toplama, tasarım, kodlama, test, konuşlandırma ve bakım olarak sıralanabilir [1]. Bu aktivitelerin her biri için geliştiricilere karşılaşacakları sorunları çözmeleri veya verimliliklerini artırmaları konusunda yardımcı olabilecek çözümler mevcuttur. Bulut bilişim, son yıllarda kişilerin kullanımına zamanla daha uygun hale gelen ve kabul gören bir teknoloji olmaya başlamış, bu kapsamda yazılım geliştiriciler için bulut tabanlı yazılım çözümlerine ve yardımcı araçlarına da daha çok rastlanmaya başlamıştır. Ancak bulut tabanlı yazılım geliştirme çözümlerinin her durum ve her proje için kesinlikle geleneksel çözümlerden daha üstün ve kullanımının daha avantajlı olduğu gibi bir genelleme yapılması doğru olmayabilir. Alternatiflerden birini kullanmadan önce projenin ve kurumun karakteristikleri, yazılım geliştiricilerin kişisel tercihleri, müşteri talepleri, ürünün ve sürecin özellikleri gibi birçok faktör göz önünde bulundurularak bir karar verilmelidir [2], böylece yanlış alternatifin seçiminden kaynaklanan zaman, efor ve para kaybından kaçınılabilir; yanlış tercihlerle verimliliğin düşürülmesi engellenebilir. Çoban vd.'nin [3] 2016 yılında Türkiye'deki yazılım geliştirme organizasyonlarının bulut kullanımı ile ilgili yaptıkları demografik ön çalışmada mevcut bulut kullanımı ile yazılım geliştiricilerin bulut teknolojilerini kullanma isteğini birlikte modelleyen istatistiksel bir çalışma yapmanın faydalı olacağı ve bulut kullanım durumunun farklı boyutları ile incelenmesi gerektiği sonucuna varılmıştır. Çoban vd.'nin [3]'te sunduğu gibi bulut teknolojilerini yazılım projelerinde kullanan organizasyonlar mevcuttur. Bu çalışmanın amacı ise Çoban vd.'nin [3] çalışmasını bir adım ileri götürerek, yazılım geliştirme projelerinde bulut teknolojileri için mevcut kullanım (MK) – algılanan uygunluk (AU) – kullanım isteği (Kİ) arasındaki ilişkinin seçilen yazılım geliştirme aktiviteleri üzerinden karşılaştırılması ve incelenmesidir. MK – AU – Kİ arasındaki ilişkilere ek olarak, bu üç kavramın farklı yazılım geliştirme aktiviteleri için sonuçlarının diğer aktiviteler ile arasında ne gibi bir ilişki olduğu istatistiksel olarak test edilmiştir. Bu çalışmanın geri kalan kısmında, Bölüm 2'de yazılım geliştirme aktivitelerinde bulut kullanımı ile ilgili bir literatür taraması, Bölüm 3'te kullanılmış olan yöntem, Bölüm 4'te istatiksel analiz ve bu analizlerin yorumları verilmektedir. Bölüm 5'te ise sonuçlar tartışılmaktadır.

2 Literatür Taraması

Yazılım geliştirme projelerinde bulut kullanımı konusuna endüstride artan bir eğilim olsa da, bu konudaki akademik çalışmalar sınırlı sayıdadır. Bulut teknolojilerinin yazılım geliştirme aktivitelerinde kullanımı için teknoloji kabul çalışmalarına, Pişirir'in [2] çalışması öncesinde rastlanmamaktadır. Bulut bilişimin kişisel kullanım için yaygınlaşmaya başladığı ilk zamanlardaki çalışmalar genel olarak bulut kavramının kendisine, yazılım alanında kullanımının olası zorluklarına ve faydalarına odaklanmıştır. Weinhardt vd. [4] bulut teknolojilerinin kullanıma hazır yazılım parçalarının erişimine olanak tanıyarak yazılım geliştirme sürecini oldukça kolaylaştıracağını savunurken; Erdoğmuş [5] bulut bilişimin yazılım geliştirme projeleri için kodlama ve test gereksinimlerini ortadan kaldıracak çapta büyük bir yenilik olacağı fikrine kuvvetle karşı çıkmıştır. Dillon vd. [6] bulut platformlarının birlikte çalışma kapasitelerinin, yazılım geliştirme alanında pratik kullanım için henüz yeterince fonksiyonel olmadığını ama ilerisi için gelecek vadettiğini belirtmiştir. Zaman içerisinde bulut teknolojilerinin yazılım geliştirme alanında oldukça büyük fayda sağlayabileceği ve bazı durumlarda geleneksel araçlardan daha avantajlı olabileceği de görülmüştür. Al-Rousan [7] 2015 yılındaki çalışmasında küresel ölçekte yazılım geliştirme alanındaki zorlukları engelleyebilmek için bulutun kullanılabileceğini önermiştir. Almorsy vd. [8] bulut ile ilgili en büyük endişelerden biri olan güvenlik kaygılarını, yazılım mühendisliği özelinde incelemiş, güvenlik önlemlerinin bulut üzerinde yazılım geliştirme eğilimleri ile birlikte nasıl gelişebileceğini yorumlamıştır. Yakın tarihli çalışmalar ise özellikle yazılım geliştirme aktiviteleri ve sorunları için bulut tabanlı çözümlerin önerilmesine odaklanmıştır. Yazılım test aktivitelerinin buluta taşınması kavramının oldukça ilgi çeken araştırma konularından biri olduğu görülmektedir. Malik ve Singh [9] çeşitli test ortamlarını tartıştıkları çalışmalarında bulut tabanlı test ortamlarını göz önünde bulundurmuştur. Benzer olarak Candea vd. [10] ise Hizmet Olarak Otomatik Yazılım Testi (*Automated Software Testing as a Service*) kavramını önermiş; Mittal vd. [11] de yazılım testlerinin geleceğinde bulut tabanlı ortamların kullanımını incelemiştir. Teixeira ve Karsten'in [12] bulut tabanlı bir yazılım ekosistemi olan OpenStack ortamı üzerine yaptıkları çalışmaları, bulutun sunduğu olanaklardan faydalanarak yazılımı erken, sık ve zamanında yayınlamayı sağlayacak bir yazılım yönetim çalışması örneğidir.

3 Yöntem

3.1 Anket Çalışması

Bu çalışma kapsamında oluşturulan MK – AU – Kİ modeline ve altı temel yazılım geliştirme aktivitesine (gereksinim, tasarım, kodlama, test, konuşlandırma ve bakım) dayalı ölçümler için 1-5 arası Likert ölçeği kullanan bir anket hazırlanmıştır. Kapsamlı anket sadece MK – AU – Kİ değişkenleri ile sınırlı olmamakta, katılımcıların proje ve organizasyon özelliklerini soran demografik bir soru setinin ardından yazılım geliştirme aktivitelerinde bulut kullanım isteğini detaylı bir model ile ölçmeyi hedefleyen birçok değişkene dayalı sorulardan oluşmaktadır. Bu değişkenler literatürde sıklıkla

rastlanan ve genel kabul görmüş teknoloji kabul teorilerinden Teknoloji Kabul Modeli (*Technology Acceptance Model, TAM*) ve Teknoloji-Organizasyon-Çevre (*Technology-Organisation-Environment, TOE*) yapılarına ilave edilen özgün bir katkı olan değişken seti Birey-Kurum-Proje (*Person-Organisation-Project, POP*) yapılarının birleşimiyle oluşturulan karmaşık bir hibrit modelin ölçümü ve hipotezlerin testleri için tasarlanmıştır [2]. Anketin tam hali (soruların tam listesi ve sunulan çoktan seçmeli seçenekler) çevrimiçi sunulmuştur [13].

3.2 İstatistiksel Analiz

Anket sonuçlarının yorumlanması ve üç değişken arasındaki farkları ölçen hipotezlerin (birinci ölçüm = ikinci ölçüm) test edilmesi sırasında üç değişken ikili olarak birbirleriyle; her değişkenin altında da altı aktivite tüm ikili ilişki kombinasyonları ile kendi içlerinde birbirleriyle, ankete verilen yanıtların *paired t-test* kullanılması ile karşılaştırılmıştır. *Paired t-test*'ler için *Student's test* veya *Wilcoxon signed-rank* gibi istatistiksel analizler kullanılabilir. Bu analizlerin Likert ölçekli anket yanıtları üzerinde uygulamasının anlamlı sonuçlar verdiği literatürde gösterilmiştir [14]. İkili karşılaştırmalarda kullanılan ve test edilen hipotez, birinci ölçüm ile ikinci ölçümün birbirinden farklı olduğu hipotezidir. p-değerinin 0.005'ten küçük olduğu karşılaştırmalarda bu hipotez %99.5 güven seviyesinde kabul edilebilir. p-değerinin 0.005'ten büyük olduğu durumlarda iki ölçümün arasında istatistiksel anlamı olan bir fark olmadığı söylenebilir. İstatistiksel analizler için bir açık-kaynaklı yazılım paketi olan JASP[1] kullanılmıştır.

4 Analiz ve Yorumlar

4.1 Veri Toplama Süreci

Türkiye'de faaliyet gösteren 30 yazılım geliştirme organizasyonuna ulaşılmış (Tablo 3), bu kurumlardan an itibariyle 84 farklı aktif yazılım geliştirme projesinde (Tablo 2) yazılım geliştirici veya proje lideri olarak çalışan 191 katılımcı (Tablo 1) seçilmiştir. Bu katılımcılara aktif olarak çalıştıkları yazılım projelerine dair bir anket uygulaması yapılmıştır. Anket uygulamaları 29/10/2017 ve 17/04/2019 tarihleri arasında organizasyonlar ziyaret edilip katılımcılarla yüz yüze yapılan görüşmelerle tamamlanmıştır. Birden fazla yazılım projesinde aynı anda çalışan yazılım geliştiriciler ile her proje için ayrı bir anket uygulaması tamamlanmış, 288 farklı gözleme ulaşılmıştır. Toplanan veri temizlendikten sonra 268 yanıt ile bu çalışmanın analizi gerçekleştirilmiştir.

Gereksinim toplama, tasarım, kodlama, test, konuşlandırma ve bakım olacak şekilde altı ana yazılım geliştirme aktivitesi için MK – AU – Kİ değişkenlerini ölçmek için katılımcılara sorulan sorular sonucunda veri seti oluşturulmuştur.

[1] https://jasp-stats.org/

Tablo 1. Anket katılımcılarının demografik bilgileri (N = 191)

Cinsiyet	***n***	***%***	***Eğitim***	***n***	***%***
Kadın	32	16.8	Lise veya ön-lisans	4	2.7
Erkek	159	83.2	Üniversite	121	63.3
			Yüksek lisans	56	29.3
			Doktora	9	4.7
Yaş	***n***	***%***	***İş Deneyimi***	***n***	***%***
18-25	20	10.4	5 yıl ve daha az	70	36.6
26-33	86	45.1	6 - 10 yıl	56	29.3
34-41	58	30.4	11 - 20 yıl	54	28.3
42 ve daha büyük	27	14.1	20 yıldan daha fazla	11	5.8

4.2 Bulgular

Anket çalışmasına katılan yazılım geliştiricilerin MK – AU – Kİ değişkenlerine yönelik verdikleri yanıtlar incelenmiş ve bu kısımda bu yanıtların özet istatistikleri ve frekansları sunulmuştur. Her yazılım geliştirme aktivitesi için MK, AU ve Kİ yanıtlarının ortalaması, standart sapması ve her yanıtın frekansı Tablo 4'te verilmiştir. Şekil 1(a) yazılım aktiviteleri için MK yanıtlarının ortalama, mod ve medyan değerlerini; Şekil 1(b) AU yanıtlarının istatistiklerini ve Şekil 1(c) Kİ yanıtlarının istatistiklerini göstermektedir.

Mevcut Kullanım (MK). Yazılım geliştiren organizasyonlarda mevcut bulut kullanımı büyük ölçüde organizasyonun üst yönetiminin aldığı kararlar çerçevesinde, müşteri talepleri ve çalışılan sektörün gerektirdikleri ile şekillenmektedir. Tamamlanan anket çalışması sonuçları incelendiğinde, yazılım geliştirme aktivitelerinden herhangi birinde bulut teknolojilerini her zaman kullandığını söyleyen katılımcıların oranı %10'u geçmemektedir.

Altı temel yazılım geliştirme aktivitesindeki MK sonuçları istatistiksel olarak birbirleri ile karşılaştırılmıştır. *Student's t-test* ve *Wilcoxon signed-rank* testleri birlikte uygulanmış, iki analizin sonuçları birlikte Tablo 5'te verilmiştir. Analiz sonuçlarının verildiği tablolarda istatistiksel olarak anlamlı derecede farklı bulunan ilişkiler koyu renkle gösterilmiştir. Yazılım geliştirme aktivitelerinde MK'nın birbiri ile olan ilişkileri incelendiğinde şu yorumlar yapılabilir:

- Gereksinim toplama, tasarım, test ve bakım aktivitelerinde MK büyük ölçüde birlikte şekillenen bir karar olarak görülebilir. Bu dört aktivite için MK değerleri arasında istatistiksel anlamlı bir fark olmadığı sonucuna ulaşılmıştır.
- Bu dört aktivite grubunun dışında, kodlama ve konuşlandırma aşamalarındaki MK diğer dört aktiviteden genel olarak bağımsız bir karar olarak verilmektedir (gereksinim toplama ve konuşlandırma arasındaki ilişki hariç).
- Kodlama ve konuşlandırma aşamalarındaki bulut kullanımı arasında ise istatistiksel bir fark olmadığı görülmektedir.

Bulut teknolojilerinin Türkiye'deki yazılım geliştirme organizasyonlarında kullanımına ilişkin karar verilirken kodlama ve konuşlandırma aktivitelerinde kullanılacak servislerin diğer dört aktiviteden bağımsız olarak verilmiş olduğu; gereksinim, tasarım,

test ve bakım aktivitelerinde ise bulut kullanımının genelde birlikte olduğu sonucuna varılabilir.

Tablo 2. Anket katılımcılarının proje bilgileri (N = 84)

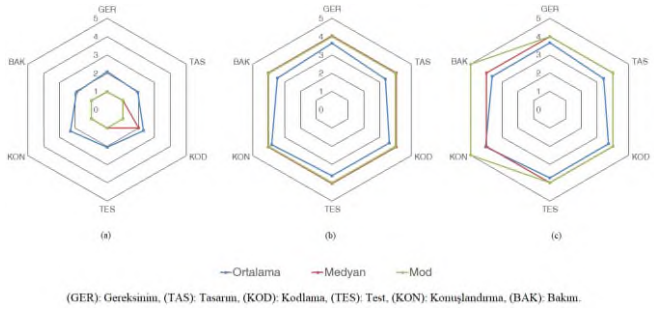
Proje Boyu	***n***	***%***	***Konuşlandırma modeli***	***n***	***%***
<10 KLOC	4	4.8	Kendi fiziksel sunucumuz	44	52.4
10-99 KLOC	13	15.5	Kiralık fiziksel sunucu	11	13.1
100-1000 KLOC	17	20.2	Kiralık sanal sunucu	20	23.8
>1000 KLOC	10	11.9	Kendi sanal sunucumuz	34	40.5
Kestirim yapılmadı	40	47.6	***Yazılım Geliştirme Modeli***	***n***	***%***
Proje Bütçesi	***n***	***%***	Çevik (Agile)	64	76.2
< $50,000	20	23.8	Yinelemeli (Incremental)	24	28.6
$50,000 - $100,000	7	8.3	Şelale (Waterfall)	14	16.7
$100,000 - $500,000	17	20.2	***Programlama Dili***	***n***	***%***
> $500,000	37	44.1	Java	51	60.7
Açıklanmadı	3	3.6	JavaScript	49	58.3
Proje Takım Boyu	***n***	***%***	PHP / ASP / JSP	29	34.5
1 – 3 kişi	24	28.6	C++	18	21.4
4 – 7 kişi	38	45.2	iOS / Swift	18	21.4
8 – 15 kişi	12	14.3	C#	16	19
16 – 35 kişi	10	11.9	Python	16	19
Yüklenici	***n***	***%***	Objective-C	11	13.1
Kurum tek yüklenici	70	83.3	Diğer	14	16.7
Konsorsiyum	14	16.7	***Mobilite***	***n***	***%***
Coğrafi Yerleşim	***n***	***%***	Hiç mobil değil	43	51.2
Bir ofis	63	75	Yarıdan azı kısmen mobil	22	26.1
Bir şehir, iki ofis	11	13.1	Yarıdan azı genelde mobil	1	1.2
Bir şehir, ikiden çok ofis	1	1.2	Yarısı kısmen mobil	13	15.5
İki şehir, birden çok ofis	6	7.1	Yarıdan çoğu kısmen mobil	1	1.2
İkiden çok şehirde ofis	3	3.6	Yarıdan çoğu genelde mobil	3	3.6
Geliştirilen Yazılım Türü	***n***	***%***	Tüm takım genelde mobil	1	1.2
Yaşam-kritik sistemler	25	29.8	***Mali kaynak***	***n***	***%***
İş uygulamaları	32	38.1	%100 yerli	70	83.3
Bilim/mühendislik	8	9.5	Çoğunlukla yerli	5	6
Sistem yazılımları	11	13.1	Çoğunlukla uluslararası	6	7.1
Web uygulamaları	52	61.9	%100 uluslararası	3	3.6
Mobil uygulamalar	30	35.7			

Tablo 3. Anket katılımcıların organizasyon bilgileri (N = 30)

Organizasyon Boyu	***n***	***%***	***Yazılım Projesi Sayısı***	***n***	***%***
1 – 9 kişi	8	26.7	1 - 10	18	60
10 – 49 kişi	9	30	11 - 25	8	26.7
50 – 99 kişi	0	0	26 - 75	2	6.7
100 – 499 kişi	7	23.3	76 - 200	0	0
500 + kişi	6	20	200 +	1	3.3
Yıllık Ciro	***n***	***%***	Raporlanmadı	1	3.3
< $100,000	2	6.7	***Organizasyon Sektörü***	***n***	***%***
$100,000 - $500,000	10	33.3	Bankacılık / finans	7	23.3
> $500,000	15	50	Kamu kurumu	14	46.7
Açıklanmadı	3	10	Askeri / savunma	11	36.7
Organizasyon Yaşı	***n***	***%***	Mühendislik / üretim	9	30
1 – 10 yıl	14	46.7	BT / telekomunikasyon	13	43.3
11 – 25 yıl	9	30	Sigorta	3	10
26 + yıl	7	23.3	Sağlık	7	23.3
			Yönetim	7	23.3

Tablo 4. MK – AU – Kİ yanıtlarının istatistikleri

	Aktivite	Ort. (x̄)	St. Sap. (s)	Hiç	Nadiren	Bazen	Sıklıkla	Her Zaman
MK	Gereksinim	2.07	1.28	122	30	39	37	9
	Tasarım	1.94	1.17	130	42	52	15	11
	Kodlama	2.30	1.44	119	29	40	38	26
	Test	2.04	1.30	131	42	34	31	15
	Konuşlandırma	2.32	1.50	126	20	28	51	25
	Bakım	1.93	1.32	144	32	28	22	18
AU	Gereksinim	3.63	1.25	27	19	31	104	61
	Tasarım	3.36	1.09	20	24	76	89	31
	Kodlama	3.63	1.15	21	16	54	101	57
	Test	3.61	1.06	13	19	65	95	49
	Konuşlandırma	3.82	1.20	19	10	39	80	76
	Bakım	3.46	1.15	20	19	56	84	37
Kİ	Gereksinim	3.64	1.31	27	20	37	78	74
	Tasarım	3.42	1.27	28	23	53	78	49
	Kodlama	3.72	1.27	23	25	37	85	84
	Test	3.73	1.15	13	26	54	82	76
	Konuşlandırma	4.04	1.17	15	12	33	68	111
	Bakım	3.63	1.23	22	12	63	63	69



Şekil 1. MK – AU – Kİ yanıtlarının istatistikleri

Tablo 5. MK analiz sonuçları

Ölçüm 1	Ölçüm 2	Student İstatistik	Wilcoxon İstatistik	df	Student p-değeri	Wilcoxon p-değeri
Gereksinim	Tasarım	1.727	1477.5	228	0.086	0.059
Gereksinim	Kodlama	-2.955	2250.5	229	0.003	0.005
Gereksinim	Test	0.227	3061	232	0.821	0.588
Gereksinim	Konuşlandırma	-1.727	3061.5	228	0.085	0.132
Gereksinim	Bakım	1.374	2029	224	0.171	0.19
Tasarım	Kodlama	-4.399	1875.5	244	< .001	< .001
Tasarım	Test	-1.104	1942	235	0.271	0.337
Tasarım	Konuşlandırma	-3.171	2342	233	0.002	0.002
Tasarım	Bakım	1.233	1814	231	0.219	0.245
Kodlama	Test	3.491	2667	242	< .001	0.003
Kodlama	Konuşlandırma	0.507	2690	235	0.613	0.564
Kodlama	Bakım	5.523	3668	229	< .001	< .001
Test	Konuşlandırma	-3.009	1272.5	239	0.003	< .001
Test	Bakım	0.685	1944	236	0.494	0.953
Konuşlandırma	Bakım	6.224	2325.5	234	< .001	< .001

Algılanan Uygunluk (AU). AU yazılım projelerinde çalışan bireylerin bulut teknolojilerinin mevcut iş ve görevlerine ne kadar uygun olduğunu düşünme seviyeleridir ve kişisel deneyim, düşünce, algı ve tecrübelerine dayanarak şekillenen bir değişkendir. Mevcut olarak çalıştıkları yazılım projesinde bulut kullanımı bu uygunluk algısını etkilemekle birlikte (kişisel tecrübeleri şekillendirdiği için), uygunluğun tek etkileyici faktörü MK olmamaktadır. Altı temel yazılım geliştirme aktivitesi için katılımcıların AU değerleri arasındaki ikili ilişkilerin istatistiksel test sonuçları Tablo 6'da verilmiştir.

MK sonuçlarındaki gruplaşmalar ile karşılaştırıldığında, bireylerin AU için farklı aktivite grupları olduğu görülmektedir. Yazılım geliştiricilerin bulut teknolojileri için AU incelendiğinde gereksinim, test ve bakım aktivitelerine bu kez kodlama aktivitesinin dahil olduğu ve bu dört yazılım geliştirme aktivitesi ile AU arasında istatistiksel bir fark olmadığı sonucuna ulaşılmıştır. Konuşlandırma aktivitesi için uygunluk sadece kodlama aktivitesindeki AU ile ilişkiliyken, tasarım için uygunluğun sadece bakım aşamasında AU ile arasında istatistiksel bir fark olmadığı görülmüştür.

Tablo 6. AU analiz sonuçları

Ölçüm 1	Ölçüm 2	Student İstatistik	Wilcoxon İstatistik	df	Student p-değeri	Wilcoxon p-değeri
Gereksinim	Tasarım	3.421	3300.5	221	< .001	< .001
Gereksinim	Kodlama	-0.337	4625	224	0.736	0.709
Gereksinim	Test	0.418	5033.5	219	0.676	0.499
Gereksinim	Konuşlandırma	-2.593	3129	208	0.01	0.016
Gereksinim	Bakım	0.943	5355.5	206	0.347	0.29
Tasarım	Kodlama	-3.775	2710	229	< .001	< .001
Tasarım	Test	-3.046	3902.5	224	0.003	0.007
Tasarım	Konuşlandırma	-4.934	2000	211	< .001	< .001
Tasarım	Bakım	-1.452	3025.5	203	0.148	0.182
Kodlama	Test	0.585	2824	236	0.559	0.501
Kodlama	Konuşlandırma	-2.713	2529	219	0.007	0.02
Kodlama	Bakım	1.326	3178.5	205	0.186	0.093
Test	Konuşlandırma	-2.949	1880	218	0.004	0.005
Test	Bakım	2.767	1678	199	0.006	0.007
Konuşlandırma	Bakım	4.307	2815	201	< .001	< .001

Kullanım İsteği (Kİ). Bireylerin yeni bir teknolojiyi alıştıkları çözümlerin yerine Kİ'ni etkileyen birçok faktör mevcuttur. Bu Kİ'ni şekillendiren sebepleri spesifik teknolojilere ve spesifik toplumlara bağlı olarak ölçmeye ve anlamlandırmaya çalışan teknoloji kabul çalışmaları literatürde rahatlıkla bulunabilen araştırmalardır. Bu çalışmada Kİ'ni şekillendirebilecek tüm faktörlerin dahil olduğu kapsamlı bir model çıkarmak yerine bu isteğin AU ve MK arasındaki ilişkiler ve farklar incelenmektedir. Katılımcıların altı yazılım geliştirme aktivitesinde bulut teknolojileri Kİ'ni ölçen anket sorularına verdikleri yanıtlar kullanılarak yapılan karşılaştırma analizi sonuçları tüm aktivite ikilileri için Tablo 7'de verilmiştir. AU ile aynı şekilde, bireylerin bulut Kİ de gereksinim, test, bakım ve kodlama aktiviteleri arasında fark göstermemektedir. Tasarım aktivitesinde bulut Kİ gereksinim aşamasında bulut Kİ ile bir ilişki gösterirken, konuşlandırma aktivitesinde bulut Kİ'nin diğer aktiviteler için olan istekten bağımsız ve farklı olduğu söylenebilir.

Tablo 7. Kİ analiz sonuçları

Ölçüm 1	Ölçüm 2	Student İstatistik	Wilcoxon İstatistik	df	Student p-değeri	Wilcoxon p-değeri
Gereksinim	Tasarım	1.619	2349	211	0.107	0.093
Gereksinim	Kodlama	-1.083	2247	223	0.28	0.199
Gereksinim	Test	-0.69	3176.5	228	0.491	0.654
Gereksinim	Konuşlandırma	-4.3	1372.5	219	< .001	< .001
Gereksinim	Bakım	-1.074	2499.5	216	0.284	0.359
Tasarım	Kodlama	-3.799	1651.5	230	< .001	< .001
Tasarım	Test	-3.674	2113	225	< .001	< .001
Tasarım	Konuşlandırma	-6.447	1238.5	213	< .001	< .001
Tasarım	Bakım	-2.244	1622	207	0.026	0.019
Kodlama	Test	-0.265	1446.5	239	0.791	0.933
Kodlama	Konuşlandırma	-3.929	578	228	< .001	< .001
Kodlama	Bakım	0.723	2633	220	0.471	0.25
Test	Konuşlandırma	-4.332	730	232	< .001	< .001
Test	Bakım	0.122	1521.5	225	0.903	0.924
Konuşlandırma	Bakım	5.418	2340	220	< .001	< .001

Mevcut Kullanım (MK) – Algılanan Uygunluk (AU). Her bir değişkenin kendi içinde yazılım geliştirme aktiviteleri üzerindeki sonuçların karşılaştırılmasına ek olarak üç kavram birbirleri ile de karşılaştırılabilir. Her bir yazılım aktivitesindeki MK ve bireylerin AU dereceleri karşılaştırıldığında ulaşılan sonuçlar Tablo 8'de verilmiştir. Altı temel yazılım aktivitesinde de MK ve AU ölçümleri arasında istatistiksel olarak anlamlı bir fark olduğu söylenebilir.

Tablo 8. MK - AU analiz sonuçları

Ölçüm 1	Ölçüm 2	Student İstatistik	Wilcoxon İstatistik	df	Student p-değeri	Wilcoxon p-değeri
MK-Gereksinim	AU-Gereksinim	-16.962	195	226	< .001	< .001
MK-Tasarım	AU-Tasarım	-18.717	273	231	< .001	< .001
MK-Kodlama	AU-Kodlama	-13.191	984	234	< .001	< .001
MK-Test	AU-Test	-16.516	315.5	227	< .001	< .001
MK-Konuşlandırma	AU-Konuşlandırma	-12.571	769	219	< .001	< .001
MK-Bakım	AU-Bakım	-14.276	150	206	< .001	< .001

Mevcut Kullanım (MK) – Kullanım İsteği (Kİ). MK oranları ile yazılım geliştiricilerin işlerinde bulut Kİ arasındaki karşılaştırmalar her yazılım geliştirme aktivitesi için yapılmış, ikili analiz sonuçları Tablo 9'da verilmiştir. MK ve AU ile benzerlik gösteren bir şekilde Kİ ve MK arasında istatistiksel farkın anlamlı seviyede olduğu görülmektedir.

Tablo 9. MK - Kİ analiz sonuçları

Ölçüm 1	Ölçüm 2	Student İstatistik	Wilcoxon İstatistik	df	Student p-değeri	Wilcoxon p-değeri
MK-Gereksinim	KI-Gereksinim	16.025	13696.5	217	< .001	< .001
MK-Tasarım	KI -Tasarım	15.979	15208	225	< .001	< .001
MK-Kodlama	KI -Kodlama	14.248	15726.5	239	< .001	< .001
MK-Test	KI -Test	18.219	18615	236	< .001	< .001
MK-Konuşlandırma	KI -Konuşlandırma	16.035	14717.5	231	< .001	< .001
MK-Bakım	KI -Bakım	16.324	11075	214	< .001	< .001

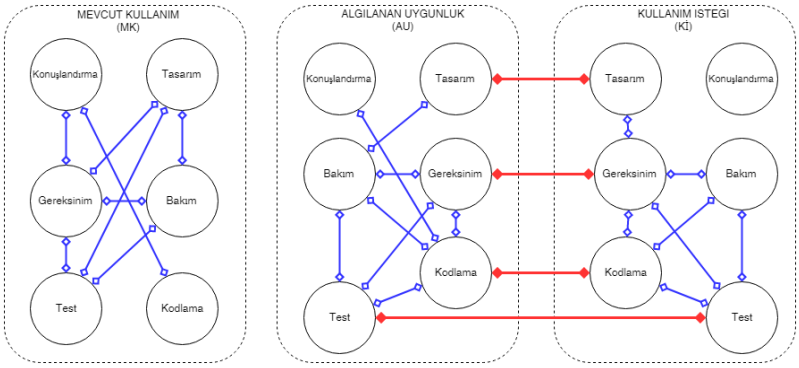
Algılanan Uygunluk (AU) – Kullanım İsteği (Kİ). Yazılım takımlarında çalışan geliştiricilerin AU ve bulutu Kİ mevcut bulut kullanımından istatistiksel olarak farklı bulunan iki değişkendir. Bu iki kavramın kendi içlerinde birbirleri ile karşılaştırılması yapılarak bu sistem hakkında sonuçlara varılabilir. Altı yazılım aktivitesi için AU ve Kİ'nin istatistiksel karşılaştırma sonuçları Tablo 10'da verilmiştir.

AU ve Kİ değişkenleri mevcut kullanımdan farklılık göstermekle birlikte, kendi aralarında altı yazılım geliştirme aktivitesinin dördü için benzer sonuçlar vermektedir. Bir yazılım projesinin gereksinim, tasarım, kodlama ve test aşamalarında çalışan yazılım geliştiricilerin AU derecesi ile bulutu Kİ arasında istatistiksel olarak anlamlı bir farka rastlanmazken, konuşlandırma ve bakım aktiviteleri için bulutu bu iki aktiviteye uygun bir teknoloji olarak görmeleri doğrudan bulutu kendileri bu aktiviteleri gerçekleştirirken kullanmak isteyecekleri anlamına gelmemektedir.

Tablo 10. AU – Kİ analiz sonuçları

Ölçüm 1	Ölçüm 2	Student İstatistik	Wilcoxon İstatistik	df	Student p-değeri	Wilcoxon p-değeri
AU -Gereksinim	KI -Gereksinim	0.762	2310	228	0.447	0.607
AU -Tasarım	KI -Tasarım	1.307	3087	225	0.193	0.23
AU -Kodlama	KI -Kodlama	1.4	2621.5	248	0.163	0.262
AU -Test	KI -Test	1.631	3106.5	234	0.104	0.057
AU -Konuşlandırma	KI -Konuşlandırma	3.331	2182	220	0.001	0.001
AU -Bakım	KI -Bakım	3.776	3102.5	209	< .001	< .001

Mevcut Kullanım (MK) – Algılanan Uygunluk (AU) – Kullanım İsteği (Kİ). Üç değişkenin birbiri ile altı yazılım geliştirme aktivitesi üzerinden karşılaştırılması ve her bir değişkenin kendi içindeki aktiviteler için değerlerinin karşılaştırılması sonucunda ulaşılan tüm model Şekil 2'de verilmiştir. Bu modelde değişkenler arasındaki oklar, aralarında istatistiksel olarak fark bulunmayan değişken ikililerini göstermektedir. MK, AU ve Kİ arasındaki ilişkiler kırmızı oklarla, her bir değişkenin kendi içindeki aktiviteler arası ilişkiler de mavi oklarla modele yansıtılmıştır.



Şekil 2. MK – AU – Kİ Modeli

Yazılım projelerinde mevcut bulut kullanımı ve yazılım geliştiricilerin bulut kullanım isteği arasında doğrudan bir bağlantı olmadığı görülmektedir. Çeşitli yazılım aktivitelerindeki mevcut bulut kullanımının genel olarak yazılım geliştirici tercihleri ile ortaya çıkmadığı, bu mevcut tecrübenin de yazılım geliştiricilerin gelecek projelerle ilişkili bulut kullanım tercihlerini doğrudan etkilemediği söylenebilir. Öte yandan dört aktivitede (test, kodlama, gereksinim ve tasarım) AU ve Kİ arasında istatistiksel olarak anlamlı bir ilişki görülmektedir. Bu aktiviteler için bulutu uygun bulan yazılım geliştiriciler pratikte de bulut servislerini kullanmaya istekli olabileceği söylenebilir. Bakım ve konuşlandırma aktivitelerinde ise yazılım geliştiricilerin bulut kullanımını uygun bulması doğrudan bu aktivitelerde bulut kullanmak isteyecekleri sonucunu vermemektedir. Aktiviteler arasındaki ilişkilere bakıldığında da gereksinim, test, kodlama ve bakım aktiviteleri için yazılım geliştiricilerin bulut uygunluk algısı ve kullanım isteği kendi içlerinde bir bütün olarak değerlendirilebilir. Geliştiricilerin konuşlandırma aktivitesi için bulut kullanım isteğinin ise diğer aktivitelerdeki tercihlerinden bağımsız olarak şekillendiği görülmektedir.

5 Sonuç

Çalışma kapsamında uygulanan anketten elde edilen veri çerçevesinde Türkiye'de yazılım geliştirme aktivitelerinde mevcut durumda bulut bilişim kullanımı oranının oldukça düşük (tüm aktiviteler için %10'dan az) olduğu görülmüştür. Bu düşük oran için yönetim seviyesinin yeni teknolojileri benimsemedeki olası tereddütleri, mevcut projeleri yeni bir ortama taşıma eforundan kaçınma isteği, askeri veya savunma sanayi gibi sektörlerdeki katı düzenleme ve kuralların bulut tabanlı servislerin alımına hiçbir şekilde izin vermemesi gibi farklı sebepler gösterilebilir. Yazılım geliştirme aktivitelerinde bulut teknolojilerini benimsemiş olan organizasyonların genelde daha az sayıda çalışana sahip olan, son yıllarda kurulmuş, daha kısıtlı bütçeye sahip organizasyonlar olduğu, teknolojik yeniliklere açık yönetim ekiplerine sahip kurumlar olduğu veya mobil uygulamalar ve sosyal medya servisleri gibi daha modern ürünler üzerine çalıştığı söylenebilecek organizasyonlar olduğu görülmüştür. Projelerde aktif olarak çalışan yazılım geliştiricilerin bulut teknolojilerini bireysel olarak işlerine uygun bulma ve bu teknolojileri kullanma isteğinin MK oranlarından daha yüksek olduğu gözlemlenmiştir. Yine de yönetimin bulut teknolojilerine sıcak bakmadığı organizasyonlarda çalışan yazılım geliştiricilerin bireysel Kİ'nde de azalma olabileceği görülmüştür. Bireysel AU ve Kİ kişiye ait deneyimler, ilgi alanları ve bilgi birikimi, tercihler, teknolojinin karakteristikleri ve çevre etkisi gibi faktörlerle şekillenebilen olgulardır. Yazılım konuşlandırma aktivitesinde bulut kullanımının veya kullanım isteğinin genel olarak diğer aktivitelerden bağımsız olduğu gözlemlenmiştir. Mevcut kullanım ele alındığında ise konuşlandırma ile birlikte kodlama aktivitesinde bulut kullanımı da diğer aktivitelerdeki bulut kullanımından farklılık göstermiştir. Bireylerin ilgi ve istekleri ölçüldüğünde ise kodlama aktivitesi diğer aktivitelerle birlikte gruplandırılabilirken, tasarım aşamasında AU veya Kİ diğer aktivitelerdeki kullanımlardan bağımsız olduğu bulunmuştur. Bulut üzerinden yazılım geliştirme yaklaşımlarının devreye alınması veya yaygınlaştırılması kararları proje ve/veya kurum yöneticileri tarafından verilirken, mevcut bulut kullanımı

yerine AU ve Kİ kategorileri ile yazılım geliştirme aktivitelerinin bu kategoriler altında kendi aralarındaki ilişkileri göz önünde bulundurulabilir. Her bir yazılım geliştirme aktivitesine özgü ve özelleştirilmiş bulut tabanlı çözümler öneren çeşitli servisler olduğundan, bu karar aşamasında yazılım geliştiricilerin proje verimliliğini artıracağına inandığı servislerin ayrıştırılması veseçimi önemlidir. Tek tek yazılım geliştirme aktivitelerinde bulut servislerin kullanılması kararı verilebileceği gibi, projedeki bir grup aktivitenin bir arada buluta taşınması da düşünülebilir. Burada dikkat edilmesi gereken nokta, söz konusu aktiviteler ile ilgili yazılım geliştiriciler tarafından AU ve Kİ değerlerinin önemli ve ilişkili olması olacaktır. Örneğin test ve kodlama gibi birbiri ile her şekilde ilişkili bulunmuş aktiviteler için birlikte çözüm öneren bulut servisleri, tasarım ve test aktivitelerine bulut tabanlı çözümler sunan bir servise tercih edilebilir.

Kaynakça

1. Bourque, P., Fairley, R. E. (Ed.): Guide to the software engineering body of knowledge (SWEBOK): Version 3.0. IEEE Computer Society Press (2014).
2. Pişirir, E.: A Structural Equation Model for Understanding the Use of Cloud by Software Development Teams (Yayımlanmamış Yüksek Lisans Tezi). Hacettepe Üniversitesi (2019).
3. Çoban, Ş., Uçar, E., Chouseinoglou, O., Sevgi, C., Testik, M. C.: Yazılım Geliştirme Takımlarının Bulut Üstünden Yazılım Geliştirme Eğilimlerinin İncelenmesi. 10. Ulusal Yazılım Mühendisliği Sempozyumu (UYMS16), Çanakkale, (2016).
4. Weinhardt, C. vd.: Cloud computing--a classification, business models, and research directions. Bus. Inf. Syst. Eng., 1, 5, 391–399 (2009).
5. Erdoğmuş, H.: Cloud computing: Does nirvana hide behind the nebula?, IEEE Softw., 26, 2, 4–6 (2009).
6. Dillon, T., Wu, C., Chang, E.: Cloud computing: issues and challenges. 24th IEEE international conference on advanced information networking and applications, 27–33 (2010).
7. Al-Rousan, T.: Cloud computing for global software development: Opportunities and challenges. Int. J. Cloud Appl. Comput., 5, 1, 58–68 (2015).
8. Almorsy, M., Grundy, J., Müller, I.: An analysis of the cloud computing security problem. Proceedings of APSEC 2010 Cloud Workshop (2010).
9. Malik, V., Singh, S.: Exploring Computing Environment Possibilities for Risk Oriented Testing. Orient. J. Comp. Sci. & Technol., 10, 3, 674-682 (2017).
10. Candea, G., Bucur, S., Zamfir, C.: Automated software testing as a service. Proceedings of the 1st ACM symposium on Cloud computing, 155–160 (2010).
11. Mittal, V., Nautiyal, L., Mittal, M.: Cloud Testing-The Future of Contemporary Software Testing, International Conference on Next Generation Computing and Information Systems (ICNGCIS), 131–136 (2017).
12. Teixeira, J. A., Karsten, H.: Managing to release early, often and on time in the OpenStack software ecosystem. J. Internet Serv. Appl., 10, 1, 7 (2019).
13. Anket Soruları, https://dar.vin/bulutanket. Son erişim tarihi: 20.08.2019.
14. De Winter, J. C. F., Dodou, D. Five-point Likert items: t test versus Mann-Whitney-Wilcoxon. Pract. Assessment, Res. Eval., 15, 11, 1–12 (2010).

Konferans, Kongre ve Sempozyum Yönetim Sistemleri İçin Bir Mobil İstemci Uygulama

Serhat Celil İleri, Selçuk Aslan ve Sercan Demirci

Bilgisayar Mühendisliği Bölümü, Mühendislik Fakültesi, Ondokuz Mayıs Üniversitesi, Samsun, Türkiye
celil.ileri@bil.omu.edu.tr, {selcuk.aslan, sercan.demirci}@omu.edu.tr

Özet. Konferans, kongre ve sempozyum yönetim sistemleri genellikle Web uygulaması olarak geliştirilmektedir. Günden güne artan mobil cihaz kullanımı bu tür sistemlerin taşınabilir cihazlarda da rahatlıkla kullanılabilir hale gelmesini gerektirmektedir. Birçok sistem bu ihtiyacı duyarlı sayfa tasarımları ve mobil cihazlara özgü sayfalar geliştirerek çözmeye çalışmaktadır. Ancak, mobil cihazlar sürekli hareket halindedir ve genellikle güvenilir internet bağlantısına sahip değildirler. Bahsedilen durumların beklenen sonucu olarak, mobil cihazların çevrimdışı inceleme ve düzenlemeye olanak sağlayan bir istemci uygulamaya ihtiyaç duydukları söylenebilir. Bu çalışmada basit düzeyde bir API sunan konferans yönetim sistemi web uygulaması ile Android işletim sistemi için geliştirilen bir istemci uygulama entegre edilmiştir. Uygulama üzerinden sisteme giriş yapılması esnasında giriş yapan kullanıcının gerekli bilgileri yerel veritabanına kopyalanmaktadır. Giriş yapan kullanıcı internet bağlantısı kesilse bile yerel veritabanındaki bilgilere erişebilmekte, düzenlemeler yapabilmektedir. Yapılan değişiklikler internet bağlantısı olması halinde doğrudan uzak sunucuya gönderilir. Aksi durumda, internet bağlantısı sağlandığında uzak sunucuyu güncellemek üzere yerel veri tabanında saklanır. Bu tür bir yöntemin uygulanmasıyla, kullanıcıların geliştirilen sistemden hem çevrim içi hem de çevrimdışı faydalanabilmesi garanti edilmiş olur.

Anahtar Kelimeler: Konferans Yönetim Sistemi, Sempozyum Yönetim Sistemi, Kongre Yönetim Sistemi, Mobil İstemci, Çevrimdışı Kullanım

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

A Mobile Client Application for Conference, Congress and Symposium Managements Systems

Serhat Celil İleri, Selçuk Aslan and Sercan Demirci

Computer Engineering Department, Faculty of Engineering, Ondokuz Mayıs University, Samsun, Turkey
celil.ileri@bil.omu.edu.tr, {selcuk.aslan, sercan.demirci}@omu.edu.tr

Abstract. Conference, congress and symposium management systems are generally developed as Web applications.The increasing use of mobile devices day by day necessitates that such systems can be used easily in portable devices. Most of systems try to solve this necessity with developing responsive page designs or mobile specific pages. However, mobile devices are constantly on the move and can not always have a reliable internet connection. As an expected result of the mentioned situations, it can be said that they require a client-side application enabling offline review and editing. In this paper, a client application developed for the Android operating system is integrated with the conference management system web application offering a simple level API. While the user tries to log-in to the system by application, necessary information are copied to local database. Logged-in user can access and edit the information in the local database even if the internet connection is lost. The changes are synchronized immediately with the remote server if there is an internet connection. Otherwise, they are stored in the local database for updating remote server when the internet connection is established. By applying this type of mechanism, it is guaranteed that users can utilize from the developed system both online and off-line.

Keywords: Conference Management System, Symposium Management System, Congress Management System, Mobile Client, Offline Usage

1 Giriş

Kongre, konferans ve sempozyumlar; bilimsel ve akademik faaliyetlerin duyurulması, tartışılması ve değerlendirilmesinde en önemli rolü üstlenen faktörler arasında gösterilebilir. Ulusal ve uluslararası olarak düzenlenebilen bu etkinlikle yurt içi ve yurt dışından bir çok bilim insanı ve akademisyen çeşitli çalışmalar ile katkı sağlamaktadır. Son derece önem arz eden ve sistematik olarak yönetilmesi gereken bu etkinlikler açısından günümüzde bilgisayar uygulamaları çok önemli bir görev üstlenmektedir.Katılımcıların kayıtlanması, çalışmaların gönderilmesi, hakem atama ve yönetimi, hakem değerlendirmeleri, çalışmanın durumunun görüntülenmesi gibi önemli işlevleri gerçekleştirebilmesi için tasarlanan konferans yönetim sistemleri gelişen teknolojiye ve yeni yönelimlere ayak uydurmaya çalışmaktadır.

Geliştirilen web uygulamaları sayesinde ulusal etkinliklerde ülkenin internet erişimi olan herhangi bir yerinden, uluslararası etkinliklerde ise dünyanın her yerindeki bilim insanları ve akademisyenlerin düzenlenen etkinliklerden haberdar olmasını sağlamanın yanısıra bu etkinliklere katılımcı olarak kaydolabilme ve çalışmalarını bu platformlar üzerinden gönderebilme imkanı bulmuş, bu sayede sürecin güvenirliği,güvenliği sağlanmakla beraber süreç hız da kazanmıştır[1]. Daha önce yapılan birçok çalışmada bu tarz etkinlikler için Web uygulamaları geliştirme sürecine [2], yöntemlerine [3] ve duyarlı tasarıma uygun hale getirme [4] konularına odaklanılmış , mobil cihazlar için istemci uygulama geliştirme ve çevrimdışı çalışılabilirlik üzerine genel olarak yoğunaşılmamıştır.

Mobil cihazların internet erişimi elde etmesi ve günden güne internete bağlanan mobil cihaz sayısının artması, internet sayfalarının ve uygulamalarının mobil cihazlar için de uygun arayüz sunma gereksinimini doğurmuştur [5]. Çok farklı cihazlar için birebir aynı görsel arayüz kullanımı özellikle küçük boyutlu mobil cihazlar için büyük bir dezavantaj oluşturmaktadır. Duyarlı (responsive) tasarıma sahip web sayfaları oluşturularak, içeriğin sayfayı görüntüleyen cihazın ekran çözünürlüğüne duyarlı şekilde oluşturulması [6], daha küçük ekranlı cihazlar için ayrı arayüzler hazırlanarak küçük ekranlı cihazlarla yapılan bağlantılarda bu sayfalara yönlendirilmesi ve uygulamasına ek olarak içerisindeki bilgileri web uygulamasından veya uzak veritabanından çekebilen mobil uygulamaların geliştirilmesi en sık başvuran çözümlerdendir [7]. Her birisinin avantaj ve dezavantajları bulunduğunu söyleyebileceğimiz bu yöntemlerin tamamının ortak yanı uygulama geliştirme ekibine değişken miktarlarda fazladan iş yükü getirmesidir.

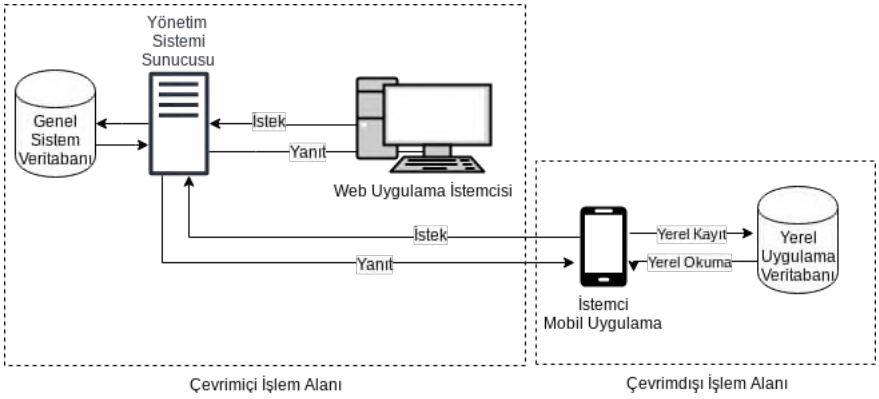
Mobil istemci uygulaması geliştirilmesi yöntemi ise, Web uygulamasından ayrı bir uygulama olarak geliştirileceği için diğer iki yönteme göre çok daha efor gerektirebilmekte olup, çoğu zaman farklı bir uzmanlık alanı olduğundan, web uygulamasını geliştiren ekipten farklı bir ekip tarafından geliştirilebilmektedir. Native uygulamalar geliştirmek için, Android platformunda Java veya Kotlin [8] dilleri, iOS için Objective-C veya Swift dillerinde geliştirme yapılması gerekmektedir [7][9].

2 Geliştirilen Uygulamalar

Bu çalışmada Giriş bölümünde mobil cihazlar için uygun arayüz oluşturmak için bahsedilen yöntemlerden sonuncusunun uygulanabilirliği araştırılmış ve olası olarak sağlayacağı getiriler ele alınmıştır. Çalışmanın gerçeklenmesi sırasında sunucu olarak işlev yürüten bir adet web uygulamasıyla birlikte bu uygulamaya dışarıdan erişimi mümkün kılan bir uygulama programlama arayüzü (API) ve istemci olarak işlev yürüten bir adet Android uygulaması geliştirilmiştir.

2.1 Web Servisi ve Uygulama Programlama Arayüzü

Uygulama Programlama Arayüzü (API), bir işletim sistemi, uygulama yada servisin özelliklerine yada verilere erişebilen uygulamaların oluşturulmasına izin veren fonksiyonlar setidir [10]. Bu çalışmada geliştirilecek mobil uygulamanın veri alışverişini sağlayabilmesi için bir web uygulaması geliştirilmiştir. Ruby on Rails (RoR) çatısı kullanılarak geliştirilen web uygulaması bir adet SQLite3 veritabanı üzerinde verilerini tutmaktadır. Geliştirilen web uygulaması ile birlikte Temsili Durum Transferi (REST) API oluşturulmuştur. Konferans, kongre ve sempozyum yönetim sistemlerinin içerdiği temel bilgiler için veritabanında ilgili tablolar oluşturulmuş ve bu veriler için oluşturma, okuma, güncelleme ve silme temel işlemlerini yapan fonksiyonlar oluşturulmuştur. API, gelen isteklere JavaScript Obje Notasyonu (JSON) biçiminde içerik ve uygun dönüş kodu ile dönüş yapmaktadır. Yönetim sistemi sunucu uygulaması, web uygulama istemcisi ve mobil istemcinin birbirleri arasındaki etkileşim ve çevrimiçi ve çevrimdışı işlem sınırları ile Şekil. 1'de gösterilmiştir.



Şekil 1. Sistemin Çevrimiçi ve Çevrimdışı Kullanım Diyagramı

2.2 Çevrimdışı Çalışma Destekli Android İstemci Uygulaması

Android Studio entegre geliştirme ortamı kullanılarak bir istemci uygulama geliştirilmiştir. Java dili ile geliştirilen uygulama, web uygulaması için bir istemci işlevi görmektedir. Web arayüzü ile yapılabilen tüm işlevleri gerçekleştirebilmesi amaçlanan uygulamada Volley kütüphanesi kullanılarak yerel sunucuda koşturulan RoR ile geliştirilen web sunucu uygulamasına Hiper-Metin Transfer Protokolü İstekleri (HTTP Requests) gönderilmektedir. Kayıtlı kullanıcı bilgileri giriş ekranında doldurularak sisteme giriş yapılabilmaktadır. Giriş işlemi esnasında sunucuya iletilen kullanıcı bilgileri geçerli ise web uygulaması tarafından kullanıcıya tahsis edilen bir token elde edilir ve uygulama önbelleğinde oturum sonlanana kadar kaydedilir. Kullanıcı tarafından gerçekleştirilecek tüm işlemlerde, giriş sırasında tahsis edilen token verisi de güvenlik ve doğrulama amaçlı olarak uygulama geliştirme arayüzü üzerinden iletilmektedir. Uygulamanın veri trafiği sadece JSON verilerinden ibaret olduğu için görsel arayüzlü bir web sayfasına bağlanmaktan çok daha az veri tüketmektedir.

Uygulamanın geliştirilmesindeki temel motivasyonlardan biri de çevrimdışı kullanabilme özelliği eklenmesidir. Kullanıcı, internet bağlantısına sahipken sisteme giriş yaptığı sırada kullanıcıyla ilişkili veriler (parola özeti ve benzeri hassas bilgiler hariç tutularak) uygulamanın yerel veritabanına çekilmiştir. Kullanıcı eğer önceden giriş yapmış ise internet bağlantısı yokken uygulamasını başlattığında yönetebildiği verilere çevrimdışı şekilde de erişim sağlayabilmektedir. Veriler üzerinde yapılan herhangi bir değişiklik durumunda uygulama içerisinde depolanan yerel veritabanı içerisinde değişiklikler uygulanarak verinin yerelde değiştiği ve internet bağlantısı sağlanması durumunda senkronize edilmesi gerektiği kaydedilmektedir. Yapılan değişiklikler, internet bağlantısı var ise hemen kaydedilmekte, internet bağlantısı yok ise bağlantı sağlandığında topluca web sunucusuna bildirilmektedir. Bu şekilde kullanıcı uygulama ile bir defa giriş yapmış ise çevrimdışı şekilde verilere erişebilir ve çevrimiçi olduğunda senkronize edilmek üzere değişiklikler yapabilmektedir. Üzerinde değişiklik yapılan veri, yerel veritabanına çekildikten sonra uzak sunucuda başka bir istemci tarafından değiştirilmiş ise öncelikle uzak sunucudaki değişiklikler kullanıcıya gösterilerek yapılan değişikliklerin uzak sunucudaki değiştirilmiş kaydın üzerine yazılmasını isteyip istemediği sorulmaktadır.

3 Sonuç

İnternet kullanımı ve internete erişim olanakları günden güne artmakta olsa da zaman zaman internet bağlantı kopuklukları veya altyapı yetersizliği olan bölgelerde internet erişiminin hiç olmaması durumu söz konusu olabilir. Konferans, kongre ve sempozyum türündeki etkinlikler özellikle yazarlar, hakem heyeti ve diğer görevli kişiler tarafından yoğun emek gerektiren etkinlikler olup, bu etkinliklere dair çalışmalar ofis ortamında olduğu kadar ofis dışında da yürütülebilmektedir. İnternet erişiminin sağlanamadığı durumlarda da bu kişilerin çalışmalarını sürdürebilmesi ve yaptığı değişiklikleri kaybetmemesi için bu sistemleri çevrimdışı olarak da kullanabilecekleri bir uygulama geliştirilmiştir.

Cihaz ile tam uyumluluk, çevrimdışı çalışabilme özellikleri ve bir grafik arayüzlü web uygulamasına oranla çok daha az veri alışverişi gerektiren bu türden bir uygulama geliştirmenin en büyük zorluğu ise tamamen yeni bir uygulama geliştirilmesi ve mevcut sistemin bu uygulamaya API sağlayacak şekilde güncellenmesidir. Ayrıca native olarak geliştirilecek uygulamalar için ise her platformda ayrı birer istemci uygulama geliştirilmesi ve yönetilmesi gerekmektedir.

Geliştirilecek bir mobil istemci uygulaması ile, bu uygulamayla uyumlu tüm konferans, kongre, sempozyum ve benzeri etkinliklerin yönetim sistemleri arasında bağlantı kurularak kullanıcıların tek bir uygulama üzerinden birden çok etkinlikteki çalışmalarını yürütebilme ve yönetebilme olanağı sağlanabilir.

4 Notlar

Bu çalışmada bahsedilen mobil istemci uygulaması, Ondokuz Mayıs Üniversitesi Bilgisayar Mühendisliği Bölümü BİL490 Yazılım Mühendisliği Laboratuvarı dersinde Final Proje Ödevi olarak verilen "Konferans Yönetim Sistemi" projesi kapsamında geliştirilmiştir. Projenin gerçeklenmesi sırasında ders kazanımlarından veri girişi, veritabanı işlemleri, uzak sunucu iletişimi ve Java dili ile Nesne Yönelimli Programlama konularından faydalanılmış ve gerçek hayat problemlerine mühendislik açısından yaklaşım konusunda deneyimler elde edilmiştir.

Kaynakça

1. Tewari, T. K., & Singh, S. K. (2010). Survey of conference management systems. International Journal of Computer Applications, 2(2), 14-20.
2. Morandini, M., vd. (2007, May). Tool-supported development with tropos: The conference management system case study. In International Workshop on Agent-Oriented Software Engineering (pp. 182-196). Springer, Berlin, Heidelberg.
3. Kanav, S., Lammich, P., & Popescu, A. (2014, July). A conference management system with verified document confidentiality. In International Conference on Computer Aided Verification (pp. 167-183). Springer, Cham.
4. Hanandeh, F. A., vd. (2015). Mobile agents modelling using UML. International Journal of Business Information Systems, 19(4), 419-432.
5. Hattori, G., Hoashi, vd. (2007, May). Robust web page segmentation for mobile terminal using content-distances and page layout information. In Proceedings of the 16th international conference on World Wide Web (pp. 361-370).
6. Gardner, B. S. (2011). Responsive web design: Enriching the user experience. Sigma Journal: Inside the Digital Ecosystem, 11(1), 13-19.
7. Serrano, N., vd. (2013). Mobile web apps. IEEE software, 30(5), 22-27.
8. Panchal, R. K., & Patel, M. A. K. (2017). A comparative study: Java Vs kotlin Programming in Android. International journal of Innovative Trends in Engineering & Research, 2(9).
9. Latif, M., vd. (2016, March). Cross platform approach for mobile application development: A survey. In 2016 International Conference on Information Technology for Organizations Development (IT4OD) (pp. 1-5). IEEE.
10. Oxford Lexico Sözlüğü, https://www.lexico.com/en/definition/api. Erişim Tarihi: 23 Haziran 2019

Nesnelerin İnterneti için Kural Denetleme Mekanizması

Cengiz Toğay[1[0000- 0001-5739-1784]], Faik Özgür[1[0000-0001-5363-5737]], Gizem Kiraz [2[0000-0002-7637-6114]] ve Durmuş Kurtuluş [2[0000-0002-1154-5300]]

[1] Uludağ Üniversitesi, Bilgisayar Mühendisliği Bölümü, Bursa
[2] EMKO Elektronik, Bursa
ctogay@uludag.edu.tr,faikozgur@uludag.edu.tr,
501631002@ogr.uludag.edu.tr,
durmus.kurtulus@emkoelektronik.com.tr

Özet. Nesnelerin interneti için geliştirmiş olduğumuz platformumuzda nesneler arasındaki iletişim içeriğinin denetlenmesi, anonimleştirilmesi ya da zenginleştirilmesi imkanına daha önceki çalışmalarımızda yer verilmişti. Mesajların içeriğine müdahale, çalışma zamanında doğrudan tatbik edilebilen Java Script tabanlı kurallar ile iletişimden sorumlu sunucuda gerçekleştirilebilmektedir. Ancak, birbirleri ile etkileşimli kural sayısının fazla olmasına bağlı olarak yöneticilerin kural tanımlamada yapacakları bir hata yeni kural ile etkileşimli tüm kuralların ve dolayısı ile kuralların uygulandığı istemcilerin etkilenmesine neden olmaktadır. Sunulan çalışmada, hazırlanan yeni kuralın gerçek hayata uygulanmadan önce test edilmesi ve sonuçlarının gözlemlenmesine imkan sağlayan yöntem sunulmuştur. Ayrıca, çalışma zamanında kuralların işlenmesi sonucunda bir mesaj hedefine ulaşamıyor ise hangi kuralın etkili olduğunun tespitinde fayda sağlamaktadır.

Anahtar Kelimeler: Nesnelerin İnterneti, Kural Tabanlı Sistemler, Benzetim.

Rule Verification Mechanism for Internet of Things

Cengiz Toğay[1[0000- 0001-5739-1784]], Faik Özgür[1[0000-0001-5363-5737]], Gizem Kiraz [2[0000-0002-7637-6114]] ve Durmuş Kurtuluş [2[0000-0002-1154-5300]]

[1] Uludağ Üniversitesi, Bilgisayar Mühendisliği Bölümü, Bursa
[2] EMKO Elektronik, Bursa
ctogay@uludag.edu.tr,faikozgur@uludag.edu.tr,
501631002@ogr.uludag.edu.tr,
durmus.kurtulus@emkoelektronik.com.tr

Abstract. In our previous study, we have developed the platform for the internet of things to control, anonymize, or enrich the communication content between things. The contents of the messages can be modified on the server responsible for communication with Java Script-based rules that can be applied directly at run time. However, due to the high number of rules interacting with each other,

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

any mistake that administrators make in rule definition affects all rules that interact with the new rule and related clients. In this study, we have presented a method that allows testing of the new rule before being applied to real-life and observing its results. In addition, this method helps to determine the cause, in this case the rule, if a message does not reach its destination as a result of processing the rules at run time.

Keywords: Internet of Things, Rule Based System, Simulation

1 Giriş

Nesnelerin Internet'i (IoT), Internet üzerinden, çok sayıda "Şeylerin", algılama, iletişim ve çalıştırma özelliklerine sahip benzersiz olarak tanımlanabilen fiziksel nesneler arasındaki ara bağlantıyı temsil eder. Internet'e bağlı şu anda 8.4 milyar akıllı nesne bulunmaktadır ve bu sayının 2020'de 20 milyar olacağı beklenmektedir [1]. "Şeyler" sınırlı hafıza, işleme kapasitesi ve enerji kaynakları gibi özelliklere sahip olduklarından Internet'e entegrasyonlarında zorluklar bulunmaktadır. IoT uygulama sayısı her geçen gün artmaktadır. Akıllı ev, sağlık izleme, akıllı şehir, kamu hizmetleri, akıllı tarım ve hayvancılık, güvenlik ve acil durumlar, akıllı su, endüstriyel kontrol, akıllı ulaşım ortamı izleme vb. bu uygulamalara örnek gösterilebilir.

IoT ortamı genişledikçe, IoT sistemlerine karşı yapılan saldırıların sayısı ve karmaşıklığı artmaktadır. IoT sistemlerine karşı yapılan saldırılar, hassas verileri çalmak, yanlış bilgi enjekte etmek veya ağların ve hizmetlerin normal işlevselliğini bozmak amacını taşımaktadır [2]. Nesnelerin interneti sistemi, farklı iletişim protokollerine sahip cihazların kullanılması, bir mesajın birden fazla istemciye aktarılması ihtiyacı, yük dengeleme, ölçeklenebilirlik ve yüksek performans gereksinimleri nedeni ile yayıncı/abone mekanizmasına göre asenkron çalışan broker'ları temel almaktadır [3]. Tüm istemciler mesajlarını broker'a gönderirler, broker, mesajı sadece konuya abone olan tüm istemcilere gönderir. Burada istemciler arasındaki anonimleştirme sağlanmış olunur. Tüm trafik broker üzerinden aktarıldığı için istemcilerin birbirlerinin IP bilgisini, kullanıcı adı gibi kimliğini ifade edecek bilgilere sahip olamazlar. Ancak, mesajın içerisinde istemciler kendileri hakkında istedikleri bilgileri paylaşabilmektedirler. Geliştirmekte olduğumuz nesnelerin interneti için güvenlik platformu kapsamında broker üzerinde değişiklikler yaparak istemciler arasındaki iletişim güvenliğini sağlamayı hedefledik. Platformda iletişimin şifreli olmasının yanı sıra verilerin belirli bir formatta olması, verilerin belirli sınırlar dâhilinde olması, anonimleştirme, zenginleştirme gibi operasyonlara imkan sağlandı [4]. Sistem, gelen mesajı konu ve gönderen bazında işleyebilmekte ve sonrasında yine kullanıcı bazında abonelerine göndermeden önce işleyebilmektedir.

IoT sistemlerine yapılacak saldırıları temelde 1) cihazın güncellenmesindeki açıklardan faydalanılarak cihaza hacker'ın uygulamasının yüklenmesi ve tüm cihazın kontrol edilmesi 2) cihazın sunucu ile olan bağlantısındaki açıklardan faydalanılarak verilerin okunması, değiştirilmesi, cihaz ya da sunucuya komut/veri gönderilmesi şeklinde

özetleyebiliriz. Platform kapsamında istemci ile broker arasındaki iletişimin güvenliğinin iyileştirilmesine yönelik yaptığımız çalışmalar ile TLS haricinde de şifreli iletişime olanak sağladık [5].

Bu çalışmamızda ise iki numaralı atak türlerine karşılık önlemler hedeflenmektedir. Cihazlar, sunuculara durumları ya da durumlarındaki değişiklikleri gönderirken, sunucudan cihazın ayarlarına müdahale edilmesi ya da belirli bir komutların cihazda çalıştırılması için gönderilir. IoT cihazlarının sınırlı bellek ve işlemci kapasiteleri nedeni ile mesajların içerik ve formatından kaynaklı hatalara karşı önlemler genelde alınmamaktadır. Buna bağlı olarak uzaktan format dışı bir mesajla cihazın çalışması engellenebilir. Daha önce yapmış olduğumuz çalışmada cihaz ya da sunuculardan gelen mesajların belirli bir formatta uygun olmasının sağlanması, olmayanların engellenmesi, mesaj içeriklerinin değiştirilmesi (anonimleştirme, veri eksiltme ya da zenginleştirme) mümkün hale gelmiştir [4]. Kurallar sisteme yüklenir yüklenmez aktif hale gelmekte ve mesajların akışını doğrudan etkilemektedir. Kurallar arasındaki etkileşim yöneticilerin özellikle dikkatinden kaçacak ve ciddi sorunlara neden olacak olayların gelişmesine neden olabilmektedir. Dolayısı ile mesajların sisteme olan etkilerinin mesajları aktif hale getirilmeden ele alınması gerekmektedir. Sunulan çalışma, hazırlanan kuralın öncelikle bir simülasyon ortamında çalıştırılması ve beklentilere uygun bir şekilde çalıştığının görülmesi üzerine sunucuda aktif hale getirilmesini hedeflemektedir.

2 Literatür ve Önceki Çalışmalar

ActiveMQ [6], açık kaynak kodlu Apache 2.0 lisansı altında geliştirilmesi devam eden bir mesaj brokerdır. Java ile geliştirilmektedir. JMS 1.1 ve J2EE 1.4 desteği sunmaktadır. Java, C, C++, C#, Ruby, Perl, Python, PHP gibi farklı programlama dilleri için uygulama geliştirme imkanı sağlamaktadır ActiveMQ, nesnelerin internet projelerinde kullanılmakta olan standart iletişim protokollerini (MQTT [7], AMQP[8], Openwire, Websocket vb.) desteklemektedir. ActiveMQ farklı protokolleri destekleyen istemcileri arasındaki protokol değişimlerini gerçekleştirebilmektedir. Broker'lar sayesinde protokol bağımsız bir iletişim alt yapısı kurulabilmektedir. Broker'lar Yayıncı/Abone (Publish/Subscribe) mekanizmasına göre çalışmaktadır. Broker'a bağlı istemciler öncelikle broker'a kullanıcı adı ve parolası ile bağlanır. Bu bağlantı neticesinde broker kullanıcıların hangi konulara yayıncı ya da abone olabileceklerini daha önce yönetici tarafından yapılan tanımlamalara göre kontrolünü sağlamaktadır. Yayıncı/Abone mekanizmasında sadece broker sistem kullanıcılarının IP'sini ve bir mesajın kimin tarafından gönderildiğini bilmektedir. Dolayısı ile abonelere belirli bir konuda bir mesaj gelir. Aboneler bu bilgilere göre hareket ederler. Burada istemciler belirli bir konuya yayınlama yetkisine sahip bir kullanıcı tarafından bir mesajın üretildiği ve broker'a gönderildiğini kabul ederler. Mesajın içerisinde varsayılan olarak kullanıcı bilgisi yoktur, eklenmiş olsa dahi herhangi kullanıcı bir başkasına ait bilgiyi mesaja ekleyip gönderebilmektedir. Dolayısı ile kaynağı abone tarafından doğrulanabilir bir bilgi değildir. Bu nedenle mesajların yayınlandıkları konular belirlenirken kullanıcıları tanımlayacak şekilde belirleme yoluna gidilmektedir. Örnek olarak "ABC/cihazId"

şeklinde konu belirlenebilir. Kullanıcı tanımlamalarına bu konuya bir kullanıcının yayıncı olabileceği ifade ediliyorsa sistem kullanıcılarının sadece yetkilendirilmiş olan kullanıcı tarafından mesajın oluşturulduğundan emin olabilirler. Herkesin yayıncı olabildiği anonim konular için hangi mesajı gerçekte kimin sisteme gönderdiği bilgisi bulunmamaktadır.

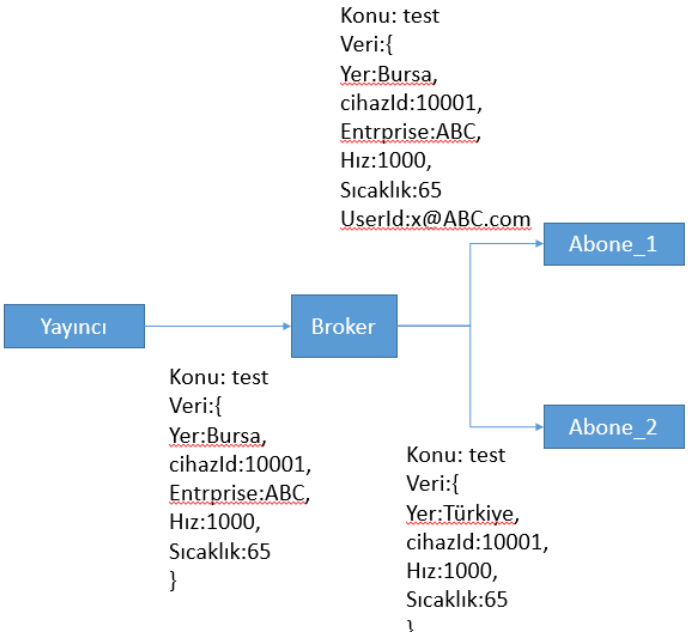
ActiveMQ üzerinde yaptığımız geliştirme çalışmaları neticesinde bir mesaj sunucuya geldiğinde ilgili kural olup olmadığının belirlenmesinin ardından yayıncı ve istemci bazında kurallar ilgili mesaja uygulanabilmektedir [4]. Böylece, broker'a dinamik kural işleme özelliği de kazandırılmıştır.

Kurallar, Javascript dili kullanılarak oluşturulmaktadır. Belirlenen koşulların sağlanması durumunda kuralların icrasının durdurulması, alınan mesaj içerinin tamamının veya bir kısmının değiştirilmesi ve değişmiş haliyle abonelere iletilmesi veya iletimin yapılmaması gibi imkanlar sağlanmıştır. Örnek olarak, Şekil 1'de gösterildiği üzere Active MQ[6] isimli broker'da topic ve/veya kullanıcıya bağlı olarak uygulanabilmektedir [4]. Sunucuya yayıncı olarak bağlı bulunan cihaz kendisine bağlı motorun hızı ve sıcaklık değeri hakkındaki bilgiyi broker'a iletmektedir. Broker daha önce cihaz kullanıcısına tanımlı bir kural var ise mesaj içeriğinin cihazdan beklenen bir mesaj olup olmadığını denetlemektedir. Sonrasında mesaj ilgili konunun üyelerine (subscriber) mesajı yine konu ve kullanıcı tanımlı kurallara göre işleyerek gönderimini sağlamaktadır. Şekil 1'de görüldüğü üzere Abone_1'e mesaj cihazın broker'a bağlanmak için kullandığı userId bilgisi ekleyerek (zenginleştirilerek) gönderilirken, Abone_2'ye "Bursa" değiştirilerek "Türkiye" şeklinde konumlandırılmıştır. Kurallar sayesinde mesaj içeriğinin değiştirilmesi ile mesaj içerisinde yer alan kişiye özel veriler kaldırılabilmekte veya genelleştirilerek (ör: kesin konum bilgisinin şehir veya bölge bilgisi ile değiştirilmesi) kişisel mahremiyetin korunması sağlayacak olanaklar sunulmaktadır.

Bir konuya abone olan cihazlar belirli tipte mesajlar almayı beklerler. Beklenmedik mesaj içerikleri veya değerleri aldıkları zaman ise sistemde hatalar veya anomaliler oluşabilir. Kurallar ile cihazın yaptığı veya yapabileceği hataları broker da engelleyip, bu hatalar yüzünden sistemin hatalı bir şekilde çalışması engellenir. Örneğin sisteme düzenli olarak 0-25 aralığında sıcaklık gelirken birden 1000 değeri gelmesi anormal bir davranışı ifade eder. Broker konuya özel tanımlanmış bir kural ile sisteme bu cihazdan gelebilecek sıcaklığı aralığını bilir ve gelen aşırı durumları sisteme eklemez, bu sayede sistem bu tip ataklara karşı korunmuş olunur.

Kurallar çalışma zamanında sisteme uygulanabilmektedir. Bir kural tanımı yapıldığında sunucuya kaydedilir kaydedilmez gelen ilk ilgili mesaja doğrudan uygulanabilmektedir. Sağladığımız bu esnekliğin kötü tarafı hazırlanan hatalı bir kuralın sistemi doğrudan etkileyebilecek olmasıdır. Hazırlanan bir kuralın mevcut kurallar ile etkileşimi de söz konusudur. Dolayısı ile kuralın mesajı etkileyebilecek sistemdeki mevcut kurallarla birlikte değerlendirilmesi gerekmektedir.

Her bir konu özelinde brokerın çalıştıracağı komutlar sıralı bir şekilde tanımlanabilmektedir. Broker ilgili konuya gelen her mesaj için sıra ve öncelik değerlerini dikkate alarak kuralları art arda çalıştırmaktadır. Kuralların çalışması neticesinde oluşan mesajın son hali abone broker kullanıcısına gönderilmektedir. Kurallar her kullanıcı için farklı olabildiği için yayıncı tarafından gönderilen bir mesaj farklı kullanıcılara farklı içerik ile gitmesi sağlanabilmektedir.



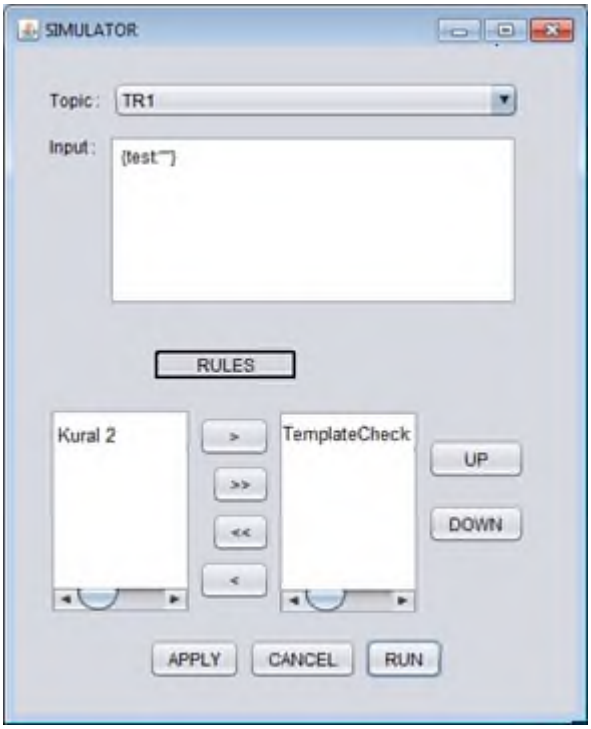
Şekil 1. Kural işleme mekanizması

3 Materyal ve Yöntem

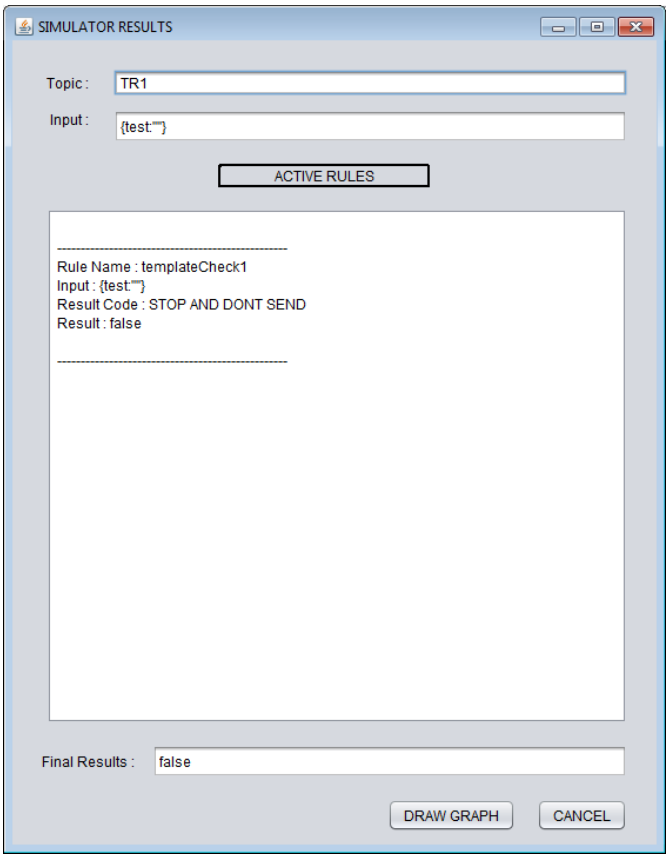
Geliştirmekte olduğumuz platform daha önce de ifade edildiği üzere mesajları işleyebilmemize olanak sağlamaktadır. Platformun işleyişi ve ekranların tasarlamasında dil olarak Java kullanılmıştır. Java Script dilinde tanımlanan kurallar yönetici tarafından platforma yüklenmektedir. Yönetici her kuralın önceliğini, hangi kullanıcıları etkileyeceğini, hangi konulara uygulanacağını sistemde tanımlamaktadır. Bu çalışmamızda hatalı kuralların çalışma zamanında sistemin işleyişine zarar vermeden etkilerini görmemizi sağlaması hedeflenmektedir. Kurallar Java Script dilinde Şekil 2 görüldüğü üzere hazırlanmakta ve sunucuya gönderilmektedir. Örnek uygulamamızda Şekil 2'de ifade edilen kural sisteme daha önce eklenmiştir. Kural gelen mesajın içeriğinde Yer, CihazId, Kurum, Hız, Sicaklik ve UserId alanlarının olmasını zorunlu kılmaktadır. Bu bilgilerden bir tanesi dahi olmazsa ya da beklenmeyen bir içerik olursa mesaj alıcılara ulaştırılmamaktadır. Örneğimizde sadece alan adları kontrolüne yer verilmiştir. Hazırlanan kuralda verilerin içeriği ve belirli sınırlar dahilinde olması sağlanabilir ve denetlenebilir. Şekil 3'de yönetim uygulamasından kayıtlı olan kurallardan Kural 1(TemplateCheck) seçilmiş ve test mesajı için değerlendirilmek üzere sunucuya gönderilmiştir. Sekil 4'de ise sunucudan gelen sonuç görülmektedir. Birden fazla kural seçilerek etkileşimlerinin görülmesi sağlanabilmektedir.

```
function GMSRuleFunction (msg){
 var isTemplateBased = utils.JSONUtility().isTemplateBased("{ Yer:"", CihazId:"", Enterpise:"", Hız:"", Sicaklik:"", UserId:""}",msg);
 if(isTemplateBased){
  return core.RuleResult(1, msg);
 }else{
  return core.RuleResult(2, isTemplateBased);
 }
}
```

Şekil 2. Şablon kontrol kuralı



Şekil 3. TemplateCheck kuralını test mesajı için değerlendirilme ekranı



Şekil 4. Şablon kuralının test mesajı için sonuç ekranı

Şekil 5'de ise hazırlanan yeni kural ile mesaj içeriğinde bir aboneye gönderilmeden önce mesajın içeriğinden UserId bilgisinin çıkartılması sağlanmaktadır. Her iki kuralda tek başına doğru bir şekilde çalışmaktadır. Ancak, hazırlanan son kural, ilkinden daha öncelikli olacak olursa Şekil 2 ile ifade edilen kurala göre mesajda UserID bilgisi olmadığı için mesajın hedefe ulaşması broker tarafından engellenecektir. Dolayısı ile cihazdan gelen bilgileri bekleyen abone uygulama ya da cihaz mesajın gelmemesi nedeni ile doğru çalışamayacaktır. Şekil 3'de gösterildiği üzere templateCheck ve Kural 2 birlikte seçilerek sunucu üzerinde çalıştırılabilir. Bu durumda öncelik sırasına göre kurallar çalıştırılacak ve sonucu yönetici tarafından görülecektir.

Hazırladığımız araç aynı zamanda bir bakım sırasında gelmeyen mesajın hangi kural tarafından nasıl engellendiğinin tespiti amacı ile de kullanılabilmektedir. Hedefe ulaşmayan örnek mesaj ve ilgili kuralların çıktıları doğrudan denetlenebilmektedir.

```
function GMSRuleFunction (msg){
  var start = msg.indexOf("UserId");
  var end = msg.indexOf(",", start)-1;
  var u = msg.slice(start, end);
  var res = msg.replace(u, "");
  return core.RuleResult(1, res);
}
```

Şekil 5. Test edilecek olan kural

4 Sonuçlar ve Tartışma

Nesnelerin interneti için cihazların ya da uygulamaların kendi aralarındaki iletişimde sadece konu ve kullanıcı tabanlı yetkilendirmenin ötesinde içerik kapsamında değerlendirme imkanını daha önce sunmuştuk. Kurallar sisteme yüklenir yüklenmez aktif hale gelmekte ve dolayısı ile şartları sağlayan ilk mesajdan itibaren uygulanmaya başlamaktadır. Sisteme yüklenecek kuralların içerik ve sayısına bağlı olarak yöneticilerin hata yapması ve sistemde beklenmeyen çıktıların oluşması bir fabrikada üretimin durması gibi durumlar ortaya çıkabilir. Hazırlanan bir kural tek başına bir mesaja uygulanmıyor olabilir. Bu durumda kurallar arasındaki etkileşim önem kazanmaktadır. Sunulan çalışmada kuralların aktif hale gelmesinden önce gerçek sistemde test mesajları üzerinden tüm ilgili kuralların çalışması sonrasında elde edilecek çıktı beklenen ile aynı olup olmadığı değerlendirilmektedir. Hazırlanan yeni kuralın neden olacağı sonuçlar, sistemin gerçek hayata uygulanması öncesinde görülebilir buna göre kuralların içeriğinin ya da önceliğinin değiştirilmesi ele alınabilir hale gelmiştir. Sunulan çalışmanın bir diğer faydası da üretilen bir mesajın hedeflere neden ulaşmadığının bulunmasında kullanılabilmesidir. Mesaj broker tarafından bir kural nedeni ile engelleniyorsa ilgili kural ve kurallar arası etkileşimin incelenmesi ile sebep bulunabilmesine olanak sağlamaktadır.

5 Teşekkür

Sunulan çalışma, TÜBİTAK 1505 (5170033) proje desteği kapsamında Bursa'da yerleşik EMKO Elektronik ile birlikte yapılan araştırma ve geliştirme çalışmalarının bir neticesi olarak ortaya çıkmıştır.

Kaynakça

1. R. van der Meulen, "Gartner Says 8.4 Billion Connected 'Things' Will Be in Use in 2017, Up 31 Percent From 2016," *Gartner Press Release*, 2017. [Online]: http://www.gartner.com/newsroom/id/3598917.
2. V. Bhuvaneswari and R. Porkodi, "The internet of things (IOT) applications and communication enabling technology standards: An overview," in *Proceedings - 2014 International Conference on Intelligent Computing Applications, ICICA 2014*, 2014, pp.

324–329.
3. D. Happ, N. Karowski, T. Menzel, V. Handziski, and A. Wolisz, “Meeting IoT platform requirements with open pub/sub solutions,” *Ann. des Telecommun. Telecommun.*, vol. 72, no. 1–2, pp. 41–52, 2017.
4. C. Toğay, “Nesnelerin Interneti için Yeni Mesaj İşleme Mekanizması,” *Uludağ Univ. J. Fac. Eng.*, vol. 23, no. 2, pp. 55–66, Jun. 2018.
5. C. Toğay, G. Mutlu, D. Kurtuluş, and F. Özgür, “Nesnelerin İnterneti için Güvenli Ağ Geçidi,” *Eur. J. Sci. Technol.*, no. 16, pp. 414–426, Aug. 2019.
6. “ActiveMQ.” [Online]: http://activemq.apache.org.
7. A. Banks and R. Gupta, “MQTT Version 3.1.1.” .
8. “ISO/IEC 19464:2014: Advanced Message Queuing Protocol (AMQP) 1.0,” 2014.

Testte Çevresel Değişikliklerin Denetiminde Olay-Yönlendirmeli İzleme Yaklaşımı

Şükrü Alataş[1] ve Aysu Betin-Can[1]

[1] Orta Doğu Teknik Üniversitesi, Enformatik Enstitüsü
Ankara, Türkiye
{sukru.alatas, betincan}@metu.edu.tr

Özet. Bu çalışmada, çevresel değişikliklerin yazılım testi esnasında nasıl denetlenebileceğini ve sistem kaynakları üzerindeki etkilerinin test kodunda nasıl teyit edilebileceğini araştırdık. Bu çalışma, popüler test çerçevelerini Microsoft Windows işletim sisteminin alt sistemi olan "Windows için Olay İzleme/Event-Tracing for Windows (ETW)“ ile birleştiren yeni bir çözüm sunmaktadır. "İzlenen Test Uzantısı/Traced Test Extension" adlı MSTest2 gibi popüler test çerçeveleriyle çalışan bir uzantı geliştirdik. Bu uzantı, işletim sisteminin test sırasındaki uygulamalar tarafından üretilen olayları dinler ve test edenin, test hedefleriyle ilgili çevresel değişiklikleri kontrol etmesi için teyit mekanizması sağlar. Uzantı, öngörülen bellek ayırma veya dosyaya, kayıt defterine veya ağa erişim gibi, işletim sistemi tarafından üretilen olayların miktarını ya da belirli özelliklerinin toplam değerini teyit etmesini sağlar. Test eden tarafından bu uzantı ile sağlanan arayüzleri kullanarak teste özel teyit kuralları da geliştirebilir. Sonuçlarımız ve vaka çalışmamız ETW ile uygulanan yöntem arasındaki işbirliğinin umut verici sonuçlarını göstermiştir. Sonuçlar ayrıca testte ETW kullanımının potansiyelini ortaya koymuştur.

Anahtar kelimeler: Yazılım Testi, Çevresel Değişikliklerin İncelenmesi

Event-Driven Tracing Approach for Inspecting the Environmental Changes in Testing

Şükrü Alataş[1] and Aysu Betin-Can[1]

[1] Middle East Technical University, Informatics Institute,
Ankara, Turkey
{ sukru.alatas, betincan}@metu.edu.tr

Abstract. In this study, we investigated how the environmental changes can be inspected in testing and how to assert the impacts on system resources in test

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

code. This study offers a new solution which connects the major testing frameworks to a subsystem of the Microsoft Windows operating system, "Event-Tracing for Windows" (ETW). We developed an extension, which works with the major testing frameworks like MSTest2, named "Traced Test Extension." This extension listens to the events produced by the operating system's event sources during testing and provides assertion for the tester to check the environmental changes regarding test objectives. The extension enables test code to check the aggregate value of the specific properties of events thrown by the operating system, such as unpredicted memory allocation and accesses to file, registry, or network. The tester could also implement custom assertion rules by using the provided interfaces in this framework. Our results and the case study showed promising results of the co-operation of the ETW and the methodology. Results also revealed the potential of ETW usage in testing.

Keywords: Software Testing, Inspecting Environmental Changes

1 Introduction

Testing is a vital software engineering process that is used for identifying whether a program or a piece of code behaves as anticipated. It is used to evaluate the conformance of software to its intended functional and non-functional requirements. Testing is usually considered as an activity to check whether the actual results match the expected results when the intended inputs are given to the software-under-test (SUT). However, testing is also dealing with many different aspects like program or system state alterations, or environmental changes during execution. These aspects should be taken into consideration to evaluate the results of the test correctly.

This study presents a new approach to automate the inspection of the environmental changes which have an essential role in assessing the results of the tests. The resulting system enables the test code to check the environmental changes during test execution, as opposed to checking as an after the fact via a profiler or other methods.

1.1 Problem Definition and Motivation

Every test has many results that no one specifies them all. “An ‘expected result’ points the tester at one or a few of these results, but it keeps away from the others [1].”

For example, suppose there is a software that would gather the keyboard inputs and write them to a file. The testing of that software may primarily evaluate whether the keyboard inputs and the resulting file match. However, the software may keep the inputs in the memory first and write them all to the file at once; or it may write every input to the file as soon as it gets them. Even if the input and the resulting file are the same, these two approaches may behave very differently in terms of the runtime environment.

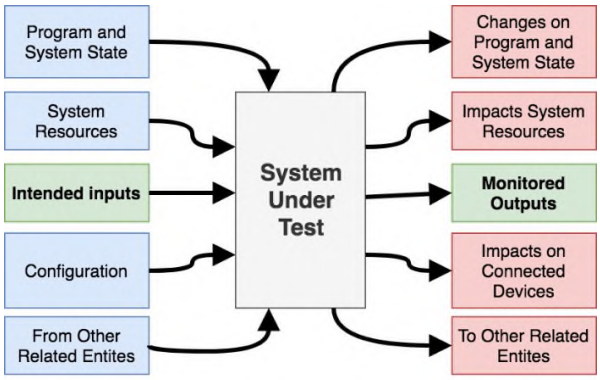


Fig. 1. Inputs and outputs of a SUT [1]

Figure 1 shows the different inputs and outputs of SUT to be examined while testing. A proper test design should cover all of them adequately to ensure correctness of the test results. Eventually, it is clear that if all we compare the test results of SUT against the expected result, we will not notice other bugs [1]. We can easily miss apparent problems. “If we are focusing our attention on something else, then we have a significant probability of completely missing an unexpected event that happens right in front of us [1].”

There is a definition of this phenomenon in psychology, which is called "inattentional blindness" [2]. When people focus on a task that demands their attention, they often fail to notice unexpected events that occur in full view. Because people typically do not consciously perceive aspects of their world that fall outside of the focus of their attention.

In software testing, there are many different inputs and outputs rather than the "intended inputs" and "monitored outputs" as defined in Figure 1. Any test process should include a thorough inspection of the results at each test [3]. To ensure the quality of the test results, all participants during the execution of the tests, not only the SUT, should be monitored and evaluated. It is necessary since the SUT may cause changes in the environment during testing. Unmonitored outputs may cause misjudgment about the test if they are not considered as an output of the SUT.

Consider our simple keyboard to file streamer application. The first approach that the software keeps the inputs in the memory first and write them all to the file at once causes memory pressure; on the other hand, the second approach that writes every input to the file as soon as it gets them causes too many accesses to the persistent storage. Even if the input and the resulting file are the same, in order to evaluate the software regarding the requirements, these environmental changes should have been observed because the software may have a performance or capacity requirements. Both of these programs conforms only one of these requirements. It is also possible that, this simple program writes the inputs 10 minutes after the last keyboard event, or that there was a memory leak and the program allocates tens of gigabytes of memory. The ‘unexpected’ impacts on system resources are also the results of that program.

In this study, we focus on a way of keeping records of the environment. We give the testers a capability of assertion on these records to validate/check the impacts of the tests on system resources with respect to the non-functional requirements of the SUT.

In this respect, we provide a new way to the tester for inspecting the impacts of the tests on the system resources. Using our mechanism, the tester can verify the SUT in different aspects, with not only the expected outputs but also the other outputs of the SUT. This also keeps tester's focus on the impacts on system resources and connected devices of a typical software test, which is shown in Figure 1.

1.2 Approach

It is apparent that we need a broader approach to observe and inspect the environment during testing. To capture all of the effects on runtime environment, the events should be traced in all levels from SUT through kernel-level. This study uses a subsystem of the Microsoft Windows operating system, "Event-Tracing for Windows" (ETW) to trace all the events from user-level through kernel-level.

Event Tracing for Windows (ETW) is a kernel-level tracing feature of the Windows operating system that lets the user observe kernel or application-defined events [4]. It was first unveiled with Windows XP operating system in early 2000. In ETW, the user can consume the events in real-time or from a log file and use them to in order to trace the internal activities of an application.

ETW has three components that are "controller," "provider," and "consumer." The controllers start and stop an event tracing session and enable providers to produce events. The providers produce events on certain conditions, like memory allocation. The consumers create a tracing session and get notified from these events when they are produced.

The diagram in Figure 2 shows the architectural model of the ETW [4].

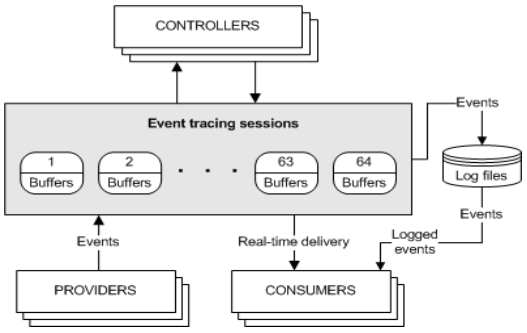


Fig. 2. Architectural model of the ETW [4]

Consumers consume the events via Event Tracing Sessions. The user creates an event tracing session, and it records the events from one or more providers. The session is also responsible for managing and flushing the event buffers.

All events have the same metadata fields that contain the necessary information shared by all event types. Moreover, events can have an additional payload to carry

more specific information about the event. The consumer can extract that information from the payload section of an event.

In this study, we propose an approach to capture events from ETW providers in a session during testing. With this approach, the tester can inspect these events to check the environmental changes regarding test objectives. The testers can also use the additional parts of the events to observe a specific type of event.

To realize this approach, we developed an extension, named "Traced Test Extension (TTE)," to popular testing frameworks for .net Framework. TTE is installed as an extension to major testing frameworks and allow testers to capture and inspect ETW events.

1.3 Related Work

ETW is heavily used primarily in performance monitoring and profiling. Currently, there are a number of tools in the market that use ETW for profiling applications and system. ETW gives the tools an ability to understand performance bottlenecks or issues. These tools simply collect and analyze the events provided.

ETW usage in software development and testing is primarily done in Microsoft Visual Studio Integrated Development Environment (IDE). Visual Studio is the primer IDE for .net Framework and Windows-oriented software development. Visual Studio has a profiling feature from the very early version, to record and visualize tracing the software development and tests. This feature uses the ETW subsystem primarily to collect the events provided by the source or test code. It collects the event and correlates them with method calls [5].

In all these tools, the effect of the program on the runtime environment is analyzed through profiling. As a general principle of these tools, the events provided by ETW are collected first, then the events refined and visualized to provide the user with an ability to understand the changes in the runtime environment. In general, they can be used in software testing as an inspection tool. Profiling a test could provide some benefits to the tester, who tries to understand the problem with the code. Finding performance problems or issues could also be useful.

However, they are general-purpose inspection tools and not specialized for software testing. The tester faces several problems when she uses these tools for inspecting the runtime environment.

The first issue is that they cannot be used in test automation easily. These tools always need a manual effort to review and understand the problem. Therefore, they cannot be used in automated testing scenarios. The automation of software testing has become a necessity, especially in agile software methodologies and test-driven development (TDD). These principles are heavily based on automated testing.

The examination of the profiling output is another issue. The review and the outputs of the review are heavily based on the tester's experience. The issues cannot be easily noticed by the tester in some circumstances. Therefore, the results of the test could be different from tester to tester.

Another issue is the output size of the profiler. In the ETW system, there could be thousands of event providers that create events in real-time. In a typical runtime environment, there could be tens of thousands of events generated in a short amount of time. A considerable portion of these events is unrelated to the SUT. However, the profiler records every event provided by the events sources in the profiling sessions. The output of the session can reach massive sizes in a long profiling session of a test. The visualization and analysis of that significant size of the output take much more system/human resources and time.

In this study, we provide a new approach to handle those problems for ETW usage in software testing. We developed an extension, which works with the major testing frameworks for .net Framework, named "Traced Test Extension."

2 Traced Test Extension Library (TTE)

In this study, we present Traced Test Extension Library (TTE) to enable the test code to examine and assert on the effect of SUT on the system resources. Traced Test Extension Library is an extension library for the major testing frameworks in .net Framework ecosystem. TTE is developed as open-source under GNU GPL v3 license that allows other entities to use, modify, and distribute the source code [6].

The source code is available at GitHub[1]. The compiled library is readily available as a NuGet package[2]. TTE is developed to target both .net Framework v4.5 and .net Standard v2.0. This selection provides the testers to use in a variety of .net Framework and other subsidiaries. The library could be used with .net Framework v4.5 to v4.8, .net Core v2.0, Mono v5.4, Xamarin, Unity and Universal Windows Platform (UWP).

TTE works with the major testing frameworks, and extend their capability to listen to the events produced by the operating system's event sources during testing and provides assertion for the tester to check the environmental changes regarding test objectives.

The extension can check the aggregate value of the specific properties of events thrown by the operating system, such as unpredicted memory allocation or/and accesses to file, registry, or network. The tester also could implement custom assertion rules by using provided interfaces in TTE. There are also sample tests provided with the source code.

Principle of TTE. TTE is designed to isolate the testing method from the environment as well as extension's procedures. There are numerous events created in the runtime environment, and the tester only needs to collect a tiny portion of them. With the help of this isolation, the tester could collect and analyze the events only produced by the test code and its related components without tackling the unrelated events produced by the other processes.

---

1 https://github.com/alatas/TracedTestExtension

2 https://www.nuget.org/packages/TracedTestExtension/

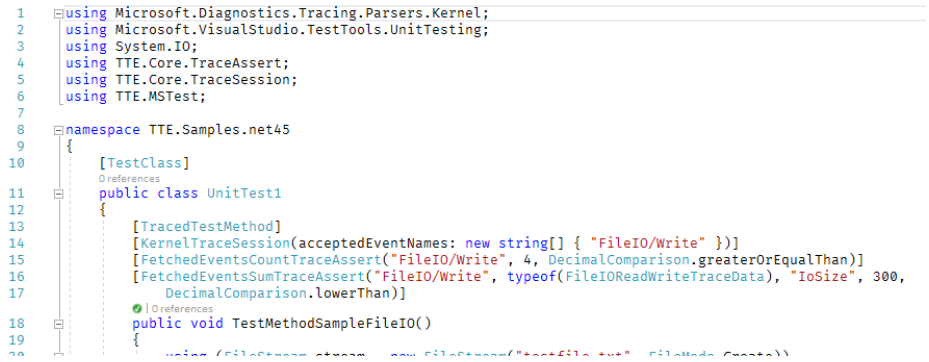


Fig. 3. An example test definition in TTE

To provide this isolation, TTE statements are written as metadata (called attribute in .net ecosystem) to test methods (refer Figure 3, line 13 – 16). The tester only needs to define these metadata annotations to use TTE. When the test driver runs the test code, TTE takes control of the execution and initiates the necessary ETW components to start the trace sessions defined in the metadata annotations of the test method. After the execution, TTE consolidates the collected events and assert the outputs with the defined assertions in the metadata annotations of the test method.

This approach provides us a clean syntax and full isolation that do not mix TTE definitions with the test codes. Otherwise, procedures of TTE might also create events which could mingle with the events caused by the actual test, causing the tester to drive unreliable results about the test.

There are three kinds of metadata annotation definition in TTE: Test method definition, Trace session definitions and Test assertion definitions.

Test method definition is a single metadata annotation to test methods. It marks the test method to be traced with the extension. The annotation we provide for test method definition is "[TracedTestMethod]." This annotation replaces the standard test method annotation (e.g., "[TestMethod]" for MSTest2) of the current test framework. The extension only traces the methods annotated with this particular annotation (refer. Figure 3 line 13). If a method is not annotated as [TracedTestMethod], then the other annotations of TTE are not evaluated by the extension.

Test session definitions are a set of specific tracing session definitions that points the individual tracing sessions the extension should create. Every tracing session has a special event parser which parses the events produced by the source. There are different types of event parser options available in ETW. Each one of them is specialized to one specific purpose of events to be collected. For example, "KernelTraceSession" includes "KernelTraceEventParser," which collects and parses the kernel-related events like file IO, memory allocation (refer. Figure 3 line 14).

Trace sessions might have a set of options which is used to filter the fetched events. The tester can filter out one of the event providers, or can only accept a set of type of events. This filtering mechanism shrinks the extension's memory footprint and speeds up the assertion process. These options can also be used to collect the events that the tester needs to collect, created outside of the testing thread. TTE has two trace session

currently, users could extend the base and create their trace sessions. In this study, we propose “KernelTraceSession” and “CLRTraceSession” annotations, which will be detailed later, one for each tracing session type. Users can also add their own annotations when they extend the base “TraceSession” class in the extension package.

Trace assert definition is a set of specialized assertion description which is used to determine the outcome of the test. The assertion procedure runs over the collected events, then decides the result of the test. There are two assertion definitions available in TTE, "FetchedEventsCountTraceAssert" and "FetchedEventsSumTraceAssert" (refer. Figure 3 lines 15, 16). Testers could extend the base class and create their custom trace assert definitions.

Predefined Trace Sessions. Currently, there are two trace sessions annotation available in TTE, "KernelTraceSession" and "CLRTraceSession." Moreover, there is also a special kind of trace session annotation, which is added by TTE to all tests, "ThreadStopKernelTraceSession."

Predefined Trace Assertions. Currently, there are two defined assertion option available in TTE which are "FetchedEventsCountTraceAssert" and "FetchedEventsSumTraceAssert". These two options could be extended through the tester's needs. The tester can inherit the base trace assertion class and write its trace assertion.

Custom Trace Assertions. When the assertion methods provided in TTE is not entirely suitable to test requirements, testers can implement their custom trace assert attribute. There is an abstract class defined in TTE named "TraceAssertAttribute" which is the base class of all trace assert attributes in TTE. There is a method in the base class named "Assert" that is used to assert the collected events. When this method is overridden, TTE provides a list of "TraceEvent" to this method after the main test method stopped.

2.1 Limitations of ETW and TTE

TTE is tightly bonded with ETW subsystem. Therefore, the limitations of ETW inherently affects TTE. Some of these limitations do not affect the daily testing routines, but some of them need to be considered before starting to use TTE.

Minor Limitations. Event Tracing supports a maximum of 64 event tracing sessions executing simultaneously in Windows Vista and succeeding Windows Operating Systems.

TTE uses ETW in real-time mode. TTE starts with 128 MB of buffer size for each tracing session. If the test system is not fast enough to handle the flowing events, the events are not captured accurately after that buffer is filled.

Major Limitations. The lack of platform support is an essential major limitation of TTE. The ETW system is currently supported only by Windows operating systems. Therefore, the TTE cannot be used in Mac OS or Linux even if the .net platform is supported on these operating systems via .net Standard targeting.

The requirement of administrative permission is another major limitation. Since administrative permission is required to create a tracing session and consume events, TTE needs to be executed by a user with administrative permission.

3 A Case Study with TTE

In this section, we provide a case study to see the feasibility and efficacy of the approach in a real-world example. The details about the case are defined first, then the problem and the solution are detailed.

3.1 Case Study Description

For the case study, we selected a long-running system which previously had an issue that can be inspected with TTE. The system is one of the busy websites which gets 80 million page views annually. The system was developed almost five years ago with .net framework v4.5.

In a technical perspective, the system is a web application which is developed in layered architecture style. The size of the system is ~80KLOC. The system has different interfaces provided to different stakeholders, such as dedicated physical devices, or different web services. All of these interfaces are connected to the same backend application for providing services.

Regarding the software testing point of view, the system has strict testing requirements for the critical components. Also, there are more than 200 integration testing methods to test the interactions between components and other systems.

3.2 Problem Description

After the deployment, an anomaly noticed about the system's memory allocation during a server inspection. The system was usually starting with memory allocation of less than 1 GB, which was quite reasonable. However, the memory usage of the application increased to 8 GB in a couple of minutes. This allocation was negatively affecting the whole system.

After a while, we found a couple of code that causes the high memory usage. We refactored those pieces, but this refactoring did not solve the main problem. There was an apparent memory leak situation that causes the problem. We focused on the “unmanaged codes,” which is the most likely cause of the memory leak problems. The code, which is developed outside .net Framework, is known as unmanaged code. The unmanaged code includes also an executable program that runs by itself. The managed code does not need to manage their resource utilization and disposal. These are coordinated by the CLR[7]. For example, Garbage Collector (GC) checks the allocated resources and disposes them. Unlike the managed code, the unmanaged code needs to manage their own resource utilization and disposal. Therefore, the unmanaged codes are the possible root cause of the memory leak problems.

After more than a week, we found the root cause of the memory leak that leads to a massive amount of memory allocation. A third-party tool developed by Microsoft causes the memory leak in a specific condition that occurred in our case. This third-party tool, which generates a file which is one of the primary outputs of the system, runs alongside the managed code in a single application domain. Therefore, the memory allocation of the tool was seen as the allocation of the whole application.

The system had already passed the unit and integration tests while we were dealing with the problem. The tests did not notice anything unexpected with the system resources because they did not inspect the runtime environment for an absolute impact caused by the code.

3.3 Application of TTE

After TTE is developed, we revisit the issue discussed above. We applied TTE on this software to find out the effectiveness and feasibility of TTE. First, we converted the "TestMethod" annotations of the MSTest testing framework to "TracedTestMethod" (refer Figure 4, line 33).

```
33          [TracedTestMethod]
34          [KernelTraceSession(acceptedEventNames: new string[] { "VirtualMem/Alloc" })]
35          [FetchedEventsSumTraceAssert("VirtualMem/Alloc", typeof(VirtualAllocTraceData),
36              "Length", 20000000, DecimalComparison.lowerOrEqualThan)]
            0 references
```

Fig. 4. The revised metadata of the test method

Then, we added "KernelTraceSession" annotations to collect kernel events. The event filter added to the annotation is "VirtualMem/Alloc" which filters the virtual memory allocation events. TTE creates a kernel trace session to collect the memory allocation events with this annotation (refer Figure 4, line 34).

Also, we added "FetchedEventsSumTraceAssert" annotation, which is used to validate the sum of a field. The annotation defined that the total value of the "Length" field of all "VirtualAllocTraceData" events should be no more than 20 MB (refer Figure 4, line 35, 36). We gathered this size measurement after the execution of the tool in a typical condition.

As a result, we defined the trace session and assertion characteristics that we need in this specific testing circumstances. Then, we run the tests as before.

3.4 Results

Before the modification, the test code verified only the monitored outputs of the test, which is a valid file with some specific properties. It could not notice the problems caused by the third-party tool who creates the output file.

After the first execution of the modified test, an assertion exception was thrown by TTE, and the test failed due to this exception. The test can quickly realize the impacts on the system resources alongside the expected output file and flagged the faulty code in a couple of seconds.

In the beginning, if we had a tool like TTE, we could easily be noticed the issues caused by the test code or third-party tools in this case. We only needed to write the expected memory allocation definitions to related test methods as annotations.

When the issue first arose, we spent more than a week to find the root cause and fix it. If we had used TTE before the issue, we would have annotated the test methods

exercising the resource critic features of the software, and we probably find the root cause in less than a week before the deployment.

4 Discussion

4.1 Findings

The case study and the other sample tests initiated with the TTE showed promising results in the inspection of the impacts on system resources in test codes. The problem with the case study was solved without any further inspection needed. We only needed to make a couple of additions to the present integration tests. The tester needed only to define the expected changes from the runtime environment. The rest was executed by the TTE. The root cause of the anomaly was revealed in a short period.

Moreover, ETW provides TTE a broader approach to inspect almost any changes in the runtime. With the help of the custom additions to TTE, the tester can track all the event providers needed and can assert the outputs easily. TTE also urge the testers to take the impacts of SUT on system resources into consideration.

However, shortcomings were also found during the study. The leading cause of these findings is ETW system, which has some drawbacks. First, the initialization and the disposal operations of the trace sessions in ETW takes a considerable time and system resources. The tests would be affected negatively by this allocation. The concurrency model of the TTE is based to minimize these adverse effects, but the overhead caused by ETW could be noticed by the tester directly. The tester might need to select some of the test cases to trace with TTE to minimize the adverse effects.

Another drawback of the ETW is the missing events. The delivery of every event provided cannot be assured by ETW. There may be some missing events that cause misjudgment about the result of tests. The missing event ratio is heavily related to system performance. If the tests are running on a system that has not enough memory, there would be more missing events.

4.2 Future work

The most critical work needed to be done in the future is supporting other popular testing frameworks like xUnit and NUnit. This expansion is supported by TTE, and these testing frameworks implicitly.

Another essential future expansion of TTE is adding other tracing session options which are available in ETW. With this expansion, the capability of the TTE will improve significantly. TTE will have the capacity to trace the webserver events, TCP/IP events, or application server events.

The inter-computer tracing is another possible future expansion of TTE. The capability of TTE will be expanded via collecting events from the computer other than the test is running. This feature may be handy for the tester when integration and system tests. These tests may cause environmental changes in other systems. These changes will be able to track with this expansion.

In connection with this expansion, the execution of the trace sessions can be departed from the extension to speed up the initialization. A separate tracker application will be responsible to collect the events. After the test, the tracker application provides the collected events to the extension. This approach may increase the performance of the initialization procedures of TTE.

5 Conclusion

Testing is the evaluation of the conformance of software to its intended functional/non-functional requirements. Testing is usually considered as monitoring expected outputs from a software-under-test (SUT) which gets the intended inputs from the tester. However, to ensure the correctness of the test results, all participants who are a part of the tests, not only the SUT, should be monitored and evaluated. A broader approach is needed to observe and inspect the runtime environment during testing. For capturing all the impacts on the runtime environment, the events created by the operating system and applications should be traced in all sub-levels from SUT to kernel-level. The resulting output of our study, "Traced Test Extension (TTE)," uses "Event-Tracing for Windows" (ETW) subsystem to give the tester an ability to inspect the runtime environmental changes during testing with ease. A feature of the Microsoft Windows operating system, ETW can trace all the events from user-level to kernel-level. In this respect, TTE alongside ETW can collect and assert almost all impacts on system resources with defined notations on supported testing frameworks. With the planned future expansions, TTE may become a unique tool to the testers to use not only for inspecting the environmental changes, but also for using as a benchmark tools, or system inspection tool during system integrations.

References

1. C. Kaner, "The Ongoing Revolution in Software Testing," in Software Test & Performance Conference, 2004.
2. A. Mack and I. Rock, Inattentional Blindness, MIT Press, 1998.
3. G. J. B. T. T. T. M. &. S. C. Myers, The art of software testing (Vol. 2)., Chichester: John Wiley & Sons., 2004.
4. Microsoft, "About Event Tracing," 31 5 2018. [Online]. Available: https://docs.microsoft.com/en-us/windows/desktop/etw/about-event-tracing. [Accessed 10 5 2019].
5. T. Arnold, D. Hopton, A. Leonard and M. Frost, Professional Software Testing with Visual Studio 2005 Team System, Wiley, 2007.
6. Free Software Foundation, "GNU General Public Licence," 29 6 2007. [Online]. Available: https://www.gnu.org/licenses/gpl-3.0.txt. [Accessed 13 5 2019].
7. Microsoft, "Common Language Runtime (CLR) overview," 2 4 2019. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/standard/clr. [Accessed 14 5 2019].

Yürütülebilir Doğal-Dil Test Gereksinimler: Bir Test-Otomasyon Deneyim Raporu

Vahid Garousi[1], Alper Buğra Keleş[2], Zeynep Özdemir Güler[2], Yunus Balaman[2]
Queen's Üniversitesi Belfast, Belfast, Birleşik Krallık[1]
v.garousi@qub.ac.uk
Saha BT A.Ş., İstanbul, Türkiye[2]
{alper.keles, zeynep.ozdemir, yunus.balaman} @sahabt.com

Özet. Test otomasyonu, endüstride sürekli büyüyen yeni metotların ve araçların kullanıldığı bir alan olmakla beraber son zamanlarda geliştirilen yeni yaklaşımlardan Behavior-Driven Development (BDD) sayesinde testlerin doğal bir dille geliştirilebilir ve çalıştırılabilir nitelikte olması sağlanmıştır. BDD yaklaşımı, test mühendislerinin doğal ve yalın bir dille test senaryoları oluşturabilmesi, iş birimleri ve yazılım ekipleri tarafından net ve anlaşılır olan bu senaryolara iş birimleri tarafından katkı sağlanması bu sayede geliştirme süreçlerinde yaşanabilen karmaşıklıkların ortadan kaldırılmasına yarar sağlamıştır. Bu bildiride, *Gauge* adlı bir test aracı kullanarak, yeni nesil otomasyon çerçevesinde yürütülebilir test spesifikasyonları geliştirerek, çalışmaların sonuçlarını bir deneyim raporu olarak sunuyoruz. Bu çalışma kapsamında, birçok ülkedeki firmalara test çözümleri sunan SAHA BT A.Ş. firmasının endüstriyel deneyimlerine yer verilmiştir.

Anahtar kelimeler: Test otomasyon, Test gereksinimler, Deneyim raporu

Executable Natural-Language Test Specifications: A Test-Automation Experience Report

Vahid Garousi[1], Alper Buğra Keleş[2], Zeynep Özdemir Güler[2], Yunus Balaman[2]
Queen's University Belfast, Belfast, UK[1]
v.garousi@qub.ac.uk
Saha BT A.Ş., Istanbul, Turkey[2]
{alper.keles, zeynep.ozdemir, yunus.balaman} @sahabt.com

Abstract. Test automation technologies are rapidly evolving and new methods and tools contantly appear the in industry. One of the new technologies is a follow-up to the Behavior-Driven Development (BDD) approach in which executable test specifications are written in the form of regular natural-language sentences and are then directly executed for the purpose of testing. Development of test suites in such a manner provides various benefits (e.g., enabling testers to write test cases in natural language) and, at the same time, exposes many research challenges which have to be addressed. We report in this paper an exploratory case-study, in the form of an experience report, with developing executable test specifications in a next generation automation framework named *Gauge*. The reported industrial experience is in the context of a large software testing company named Saha BT A.Ş. which provides automated testing tools and services to a large number of clients in several countries.

Keywords: Test automation, Test specifications, Experience report

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

1 Giriş

Yazılım testi maliyetli ve çaba gerektiren bir iş sürecidir. 2013'de Cambridge Üniversitesi tarafından hazırlanan bir rapora göre [1], 2013 itibari ile küresel ölçekte yazılım hatalarının bulunması ve çözümlenmesi toplamda 312 milyar dolarlık bir masrafa mal olmaktadır.

Yazılım test işlemleri genellikle manuel (el yordamı) ve otomatik (test otomasyon) olmak üzere iki şekilde gerçekleştirilmektedir. Manuel test yaklaşımında, testçi (test mühendisi) son kullanıcının rolünü alarak, Test-Edilen Yazılımı (TEY) ile kılavuz ile birlikte klavye ve fare kullanarak etkileşim yapmakta ve TEY' in her test durumunu (test case) doğru yapıp yapmadığını kontrol etmektedir. Bir diğer yöntem olan, otomatik test yaklaşımında ise test mühendisi özel bir test aracı (yazılım) kullanarak (örneğin: JUnit test çerçevesi), test kodu geliştirip böylece geliştirilen test kodlarını TEY üzerinde çalıştırırlar. TEY ile etkileşim, insan yerine test aracı ile yapılarak test işleminin hızı, verimliliği ve dikkati artırılabilmektedir. [2-4]. Bu konu üzerine örnek bir endüstriyel vaka-çalışması (case-study) raporunda [5], uluslararası büyük bir yazılım firmasının toplam test masraflarında, test otomasyonunun bir yılda 3.2 milyon dolar tasarruf sağladığı belirtilmiştir.

Test otomasyonu teknolojileri dünyada [6]ve Türkiye'de [7] eş zamanlı olarak hızlı bir şekilde gelişmektedirler. Bu gelişen teknolojilerden biri de Davranışa-Yönelik Geliştirme (DYG), *İngilizce: Behaviour-Driven Development (BDD)'*dir. BDD' de test durumları doğal dilde (örneğin: Türkçe veya İngilizcede) yazılıp ve koşulabilmektedirler. Bu yöntemin İngilizce metodolojisinde üç anahtar kelimesi mevcuttur bunlar; "*Given, When, Then"* şeklindedir. BDD' nin Türkçeye de aktarılmasıyla [8] yukarı da belirtilen üç anahtar kelime sırasıyla "*Diyelim ki, Eğer ki, O zaman"* şeklinde kullanılmaktadır.

BDD yaklaşımı faydalı olmasına rağmen, birçok test mühendisinin yorumlarına göre [9, 10], BDD'nin üç-kelime yapısı bazı durumlarda sınırlayıcı olabilmekte ve her test durumunu geliştirmek için uygun olmayabilmektedir, buna istinaden Gauge adlı test aracının ortaya çıkarılma sebebi bu şekilde anlatılmıştır: "*Unlike BDD tools, Gauge does not prescribe the process with a strict syntax*" [9], tercüme: "*BDD araçlarından farklı olarak, Gauge, test işleminin sınırlayıcı bir yöntem ile reçete etmez*".

Diğer bir yandan, farklı çalışma[11] ve makalelerde de belirtildiği üzere [12], yazılım gereksinimleri çoğu zaman doğal dilde yazılmaktadırlar ve çoğu firma test durumlarını doğal dilde yazılmış olan gereksinimlerden manuel şekilde tasarlamaktadır. Bu işlem genelde çok maliyetli, çaba-gerektiren ve hata-içerebilen bir süreçtir. Bu sebeplerden dolayı, doğal-dil işleme (*İngilizce:* Natural Language Processing (NLP)) kullanılarak bu alanda yayınlanmış çok sayıda çalışma ve yöntem bulunmaktadır [13]. Ancak, doğal dilde yazılmış gereksinimlerin özgür yazı tarzlarını kullanması için, yöntemlerin kapsamı genelde 100%'e ulaşamamaktadır. Daha da kötüsü, eğer gereksinimler beklenilen gramer ve formda yazılmamışlarsa, doğal-dil işleme yanlış test durumları üretebilmektedir [13].

Yukarıda belirtilen tüm sıkıntıları çözmek adına, 2018 yılında, dünyanın ünlü yazılım firmalarından, ThoughtWorks, *Gauge* adlı bir test çerçevesi tasarlamış ve açık-kaynak kodlu olarak kullanıma sunmuştur. Gauge yazılım gereksinimlerinin doğal dil şeklinde yazılmalarını sağlayarak bu gereksinimlerin direkt bir şekilde yürütülmelerini sağlamaktadır. Böylece, Gauge test otomasyonu bir sonraki seviyeye yükselmektedir. Ayrıca Gauge'e benzer mevcut hiçbir test aracı bulunmadığı için yeni nesil test otomasyon aracı olarak adlandırılabilir.

Türkiye'nin lider konumunda bulunan yazılım ve test servis firmasında (Saha BT A.Ş.) gerçekleştirilen sistematik bir analiz sonucunda [14], büyük web ve mobil uygulamalarının testi için, Gauge test çerçevesi seçilmiş ve birçok sayıda test modülü geliştirilmiştir.

Bu bildiride sunulan çalışma, bir endüstri-akademi iş birliği [15-17] olmakla birlikte "TESTOMAT – The Next Level of Test Automation" başlıklı EUREKA ITEA3 projesi çerçevesinde gerçekleştirilmiştir. Amacımız, Gauge'i Saha BT'nin seçilmiş belirli birkaç test projesinde sistematik şekilde kullanmak, test otomasyonun etkisi ve faydalarını değerlendirmek ve deneyim raporu şeklinde keşifsel bir vaka-çalışması (*İnglizce:* exploratory case-study) [13] sunmaktır. Test otomasyonun etkisi ve faydalarının analizi amaçlı literatürde birçok makale yayınlanmıştır [18-21].

2 İlgili çalışmalar

Test otomasyonu hem endüstri [22] hem de bilimsel literatürde [3] aktif olarak rağbet gören bir konudur. Bu çalışma kapsamında da belirli seçilmiş örnek makaleler üzerinden ilerlemeyi planladık [6, 20, 23-26].

Test otomasyonu alanında birçok endüstriyel vaka-çalışmasının toplandığı ve sunulduğu bir kitap literatürde mevcuttur [6]. Diğer yandan, BDD test yaklaşımı kullanılarak, birçok çalışma yapılıp makalelerde sunulmuştur [23, 24].

Bu bildirinin ilk yazarının da bu alanda birçok çalışması bulunmaktadır ve "Türkiye de gömülü yazılımlar için test otomasyon deneyimi" [25], havacılık yazılımların test otomasyon deneyimi" [26], ve "hukuk yönetim yazılımların test otomasyon deneyimi"[20] adlı çalışmalar bunlardan bazılarıdır.

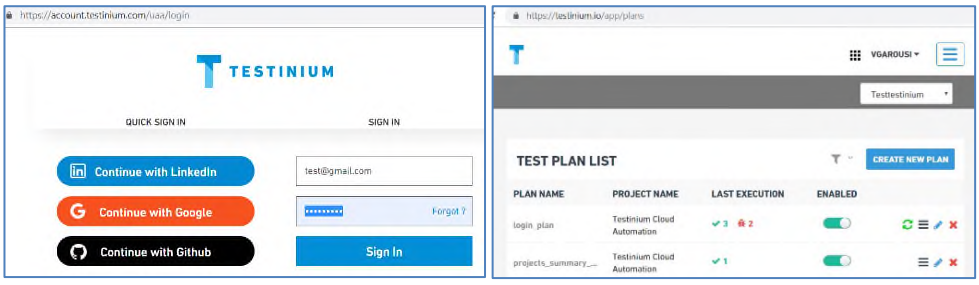
Kanıta-dayalı yazılım mühendisliği (*İnglizce:* Evidence-based software engineering) [27] alanında yer alan bu makale, direkt yürütülebilir doğal-dil test gereksinimleri ile test otomasyon deneyimi ve sonuçlarını sunmaktadır.

3 Endüstriyel bağlam ve Ar-Ge projesinin amacı

Saha Bilgi Teknolojileri (BT) A.Ş. 2010 yılında yazılım projelerine danışmanlık, eğitim ve yazılım geliştirme amacı ile kurulmuştur. Kısa sürede Java konusunda deneyimli bir ekip kurularak Türkiye'nin önde gelen teknoloji firmalarına hizmet vermeye başlamıştır. Zaman içinde Yazılım Test Otomasyonu alanında uzmanlaşarak endüstri tarafından kabul edilen Selenium ve Appium açık kod platformlar üzerinde çalışan bir test otomasyon ürünü geliştirmeyi iş planlarına dahil etmiştir. 2014 yılında bir TEYDEB projesi çerçevesinde, kendi Ar-Ge ekibini kurarak global pazarda rekabet edebilecek bir test otomasyon ürünü üzerinde çalışmaya başlamıştır, şu anda firma bünyesinde 183 personel bulunmaktadır. Ar-Ge projeleri sonucu geliştirdiği ürünleri: Testinium, Loadium, S-Box, QA Dashboard ve Analytics'dir.

Test otomasyon hizmetleri ve test araçları sunan bir firma olmasından dolayı, Saha BT her zaman daha etkin ve etkili yeni test yaklaşımlarını aramak ve geliştirmeyi hedeflemektedir. Çoğu test aracı ve çerçeveleri endüstride mevcut durumda bulunduğu için, Saha BT'nin test mühendisleri belirli her bir test otomasyon ihtiyacı için, literatürde bulunan[14] mevcut sistematik yöntemleri kullanarak, en doğu test aracını seçmektedirler.

2018'de endüstride pazara çıktığından bu yana, test araçları çeşitli projelerde pilot şekilde kullanılıp başarılı sonuçlar alındıktan sonra, 2019 yılı ikinci dönemi itibari ile daha fazla projede kullanılmaya başlanacaktır. Bu test araçlarının kullanıldığı projelerin birisi de firmanın kendi test aracı olan Testinium'un kendisini test etmesi diğer bir deyişle, test aracını test etmektir. Bu makalenin sonraki kısımlarında (Başlık 4.2) bu test-edilen yazılım (TEY) dan bahsedeceğimiz için, kısa şekilde Testinium'u açıklarsak; Testinium web-tabanlı bir araç olarak, Gauge, Seleinum gibi birçok test aracında yazılan test kodları üzerine test yönetim işlemlerini sağlamaktadır. Şekil 1'de Testinium ara-yüzüne ait bir ekran görüntüsü verilmektedir.



Şekil 1. Testinium test aracını Gauge'de yazılan test kodları ile test etmek. İlk ekran görseli login sayfasını göstermektedir.

Makalenin bütünlüğünü korumak ve anlaşılabilirliğini artırmak adına, yürütülebilir test gereksinimleri konusu üzerinden bir örnek ile ilerleyebiliriz. Testinium'un login sayfasına yazılan örnek bir test Tablo 1'de gösterilmiştir. Bu test kodlarında önemli olan noktalardan biri soyutlama seviyesidir (*İngilizce: level of abstraction*). Görüldüğü üzere, Gauge'de yazılan test gereksinimleri yüksek soyutlama seviyesine sahiptirler ve kodlar arası fonksiyon çağrıları devam ederek Java'da yazılmış Seleinum test kodları çağırılmakta ve son olarak test-edilen yazılımı (TEY) çağırmaktadır. Basit gibi görünen bu faydalı mantık ilişkisi Gauge'in en güçlü özelliğidir ve akademik literatürde mükemmel seviyede çözülemeyen bir soruyu açıkça çözümlemiş olmaktadır [13]. Tablo 1'de verilen terimleri (test gereksinimleri vb.) bir sonraki başlık altında açıklayacağız.

Bu form kapsamında test kodu yazmak, soyutlama seviyesini adım adım değiştirmektedir ve makalenin sonraki kısmında tartıştığımız birçok faydayı test mühendislerine sağlamaktadır. Bu çalışmanın gerçekleştirildiği Ar-Ge projesinin amacı Gauge'i Saha BT'nin çeşitli test projelerinde sistematik şekilde kullanmak ve test otomasyonun etki ve faydalarını değerlendirebilmektir.

Tablo 1. Testinium'un login sayfasına yazılan örnek bir test

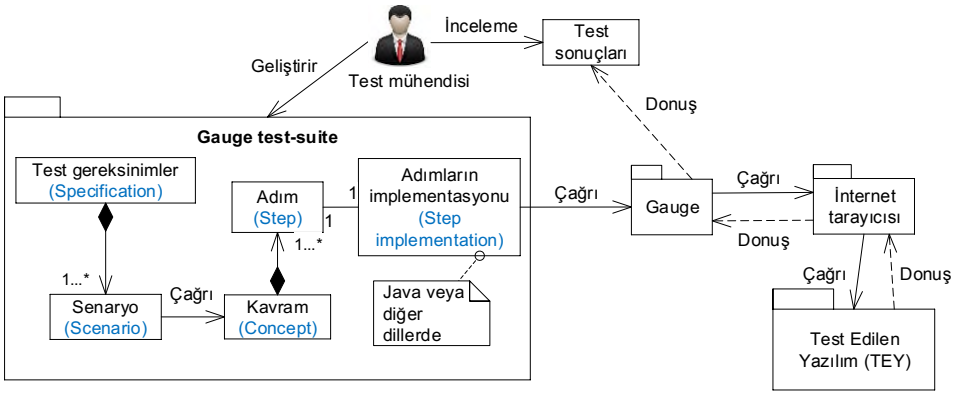
<table>
<tr><td rowspan="2">Soyutlama seviyesi (level of abstraction): yüksek</td><td>1 test gereksinimi (specification). 3 test senaryosu</td><td>Tags:Login_InputAlanKontrolleri
*Login ekranındaki Email alanının kontrollerini yap
*Login ekranındaki Password alanının kontrollerini yap
*Login ekranındaki LinkedIn butonunun kontrollerini yap</td></tr>
<tr><td rowspan="2">2 test kavramı (concept). 3+4 test adımı (test steps)</td><td># Login ekranındaki Email alanının kontrollerini yap
*"https://***.testinium.com/***/***" adresine git
*"tbEmailOfLogin" elementinin görünürlüğü "true" mu kontrol et
*"testinium***" ve "***" ile giris yap</td></tr>
<tr><td rowspan="2">Soyutlama seviyesi: düşük</td><td># <email> ve <password> ile giris yap
* "tbEmailOfLogin" elementine <email> değerini yaz
* "tbPasswordOfLogin" elementine <password> değerini yaz
* "btnSignInOfLogin" elementi var mı
* "btnSignInOfLogin" elementine tıkla</td></tr>
<tr><td>Bir test adımın implementasyonu (step implementation)</td><td>@Step("<key> elementine <text> değerini yaz")
public void sendKey(String key, String text){
ElementInfo elementInfo = StorHelper.INSTANCE.findElementInfoByKey(key);
sendKeyBy(ElementHelper.getElementInfoToBy(elementInfo),
elementInfo.getIndex(), text);
}</td></tr>
</table>

4 Test otomasyon yaklaşımı ve uygulanması

Bu ana başlık altında öncelikle test otomasyon strateji ve mimarisinden, daha sonra test kodlarının geliştirilme sürecinden ve son olarak da test kalıplarının kullanımı ve test kokularının önlenmesinden bahsedilecektir.

4.1 Test otomasyon strateji ve mimarisi

Test otomasyonun başarılı olması için, uygun ve etkin bir test strateji tasarlamak gerekmektedir [28]. Bu çalışmada siyah-kutu (black-box) ve arayüz-bazlı (GUI-based) test yaklaşımı ile ilerlemeyi planladığımız için, test aracı olarak Gauge seçilmiştir. Şekil 2' de Gauge ile belirlenen test otomasyon test mimarisini göstermektedir.



Şekil 2. Test otomasyon mimarisi

Test mühendisi tarafından Gauge için test kümesi geliştirilmektedir. Etkin test-kod geliştirilmesi için [4] (örneğin: testlerin kullanılabilirliği), Gauge uygun test tasarım özellikleri sağlamaktadır. Tablo 1'de verilen örnek kod göz önüne alınarak, bu kümede ilk önce test gereksinimleri (specification) geliştirilmektedir.

Test gereksiniminde, siyah-kutu (black-box) bir test durumudur ve TEY'in özelliklerinin dokümantasyonu da algılanabilmektedir [29]. Bir test gereksinimin içinde bir veya daha fazla test senaryoları yer almalıdır. Her senaryo ile eşlenen bir test kavramı (concept), normal kod fonksiyonlarına benzer olarak geliştirilmelidir (Tablo 1'deki örneğe bakınız). Her test kavramı içinde bir veya daha çok test adımı yerleşebilir. Son olarak, bu adımları yürütülebilir hale getirmek için, adımların implementasyonunun Java veya diğer dillerde geliştirilmesi gerekmektedir. Örneğin, Tablo 1'de `"<key> elementine <text> değerini yaz"` test adımı için, Java dilinde Selenium'u kullanarak TEY'de `<key>` elemanı bulup ve onun içinde `<text>` değerini yazıyoruz.

Görüldüğü üzere, test kavramlarının (concept) yeniden kullanılabilirliği amacıyla adımları mantıksal gruplandırarak tek bir ünitede birleştirme olanağını sağlamaktadır. Yukarıda bahsedildiği gibi farklı soyutlama seviyeleri bir araya gelerek yüksek soyutlama seviyesinde test gereksinimleri yazarak, direkt yürütmeyi (koşmak) mümkün kılmaktadır.

4.2 Test-durum tasarımı ve test-kod geliştirilmesi

Test-durum tasarımı, yazılım test alanında önemli bir yere sahiptir [30]. Siyah-kutu ve beyaz-kutu test-durum tasarım tekniği gibi birçok yöntem mevcuttur. Keşifsel vaka-çalışmamızda, yazılım gereksinimleri kullanılarak, denklik-sınıfı bölümleme (equivalence-class partitioning) tekniğini kullanarak, test-durumları tasarlanmış ve Gauge kodları geliştirilmiştir. Vaka-çalışmamız firmanın mevcut üç test projesine odaklanmıştır. Tablo 1'de belirtilen üç TEY listesini ve Gauge kodlarının ve boyut ölçütlerini göstermektedir. Bilgi gizliliği ve korunumundan dolayı, Testinium hariç diğer sistemlerin isimleri makale kapsamında verilmemektedir, yazılım tipi/ sektör sütunu gibi anonim adlar (TEY2 ve TEY3) gösterilmektedir ve bunlar Türkiye de büyük kullanıcıya sahip büyük ölçekli e-ticaret ve telekomünikasyon sistemleridirler.

Test kodlarının geliştirilme sürecinde en kaliteli test-kodlarını elde etmek için endüstri ve akademinin en iyi pratikleri (best practices) oldukça kullanılmıştır [4]. Bir sonraki başlık altında bu konu detaylı sunulacaktır.

Tablo 1. Farklı test-edilen yazılımlar (TEY) için test ve kod istatistikleri

Test-edilen yazılım (TEY)	Yazılım tipi/ sektörü	Ekran (özellik) sayısı	Gauge test gereksinim sayısı	Test senaryo sayısı	Test kavram sayısı	Test adım sayısı	İmplementasyon Java kod satır sayısı	Efor tahmini (adam ay)

TEY1-Testinium	Test otomasyon aracı	7	162	1,667	518	2,658	1,123	4
TEY2	E-ticaret	14	114	539	67	449	3,415	12
TEY3	Telekomünikasyon	8	14	181	5,411	508	1,348	5

4.3 Test kalıplarının kullanımı ve test kokularının önlenmesi

Tasarım kalıpları (design patterns) tanımından ilham almış olan test kalıpları (test patterns) [31, 32] üst-seviye test otomasyon pratiklerin biridir. Test kalıpları yazılım test projelerindeki genel problemleri en iyi yolla çözmeyi planlamaktadır. Birçok test kalıbı literatürde de mevcuttur [31], örneğin: dört-adım (four-phase) test kalıbı bir test metodun dört-adıma bölünmesini önermektedir: kurulum / hazırlama (setup), egzersiz (exercise): TEY veya birimi çalıştırmak, doğrulama (verify), ve test-sonu temizleme (teardown) şeklindedir. Daha önceki projelerimize benzer şekilde [20, 26] bu projede de test kalıpları kullanılmıştır. Uyguladığımız iyi pratiklerin bir diğeri ise test kodlarının modüler ve kolayca anlaşılabilirliği ile kolay bakımıdır.

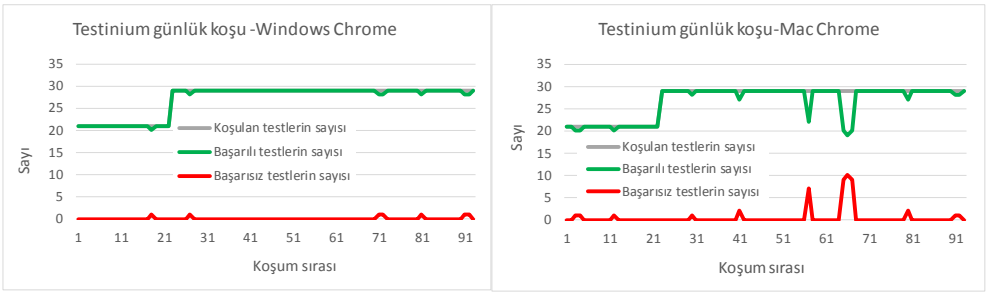
Ayrıca, test otomasyon dünyasında iyi pratiklerin tersi anlamına gelen anti patern veya test kokuları (test smells) terimleridir [33], örneğin: test-kod duplike olması test-kokusu ve karmaşık test mantığı test-kokusu gibi. Projemizde test-kodları geliştirilirken, bu ve diğer test kokularını mümkün oldukça önledik ve kodda bulduğumuzda da düzeltme yaptık (*refactoring*).

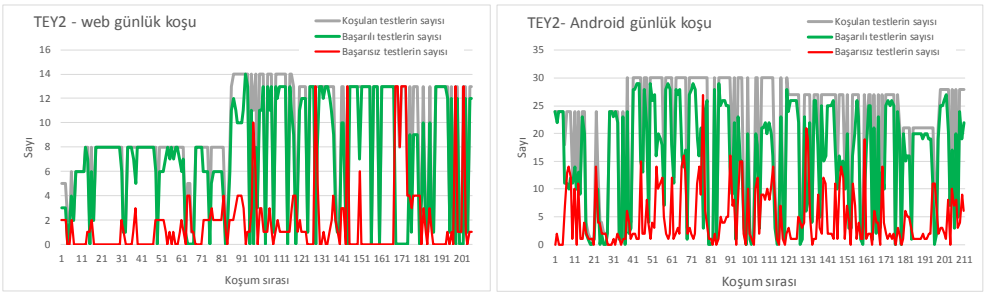
Özet olarak, test kalıplarının kullanımı ve test kokularının önlenmesi kaliteli test-kodları geliştirebilmek adına yardımcı oldular. Sonraki makalelerimizde, bu konularda daha detaylı örnek ve bilgileri paylaşıyor olacağız.

5 Test otomasyonun etkisi ve faydaları

Test ve test otomasyonun amacı TEY' de mevcut olan hataların hepsini bulmak veya sisteme olan güveni arttırmaktır [30]. Test otomasyonun etkisi ve faydalarını değerlendirmek için, birçok yöntem kullanılmaktadır bunlar: (1) testlerin TEY' in geçen sürümlerinde bulduğu hataların sayısı ve trendi, (2) TEY kod ve gereksinimlerin kapsama oranı (test coverage) ve (3) mutasyon testi (TEY' e bilerek hata enjekte etmek) şeklindedir. Başlangıç adımı olarak ilk yöntem bu projede uygulanmıştır. Seçtiğimiz TEY'lerin (Tablo 1) ikisi için (Testinium ve TEY2) ve ayrıca her bir sistem için iki farklı test konfigürasyonu, Şekil 3'de de Gauge testlerin belirli bir zaman döneminde hata bulmak trendi görselleştirilmiştir. DevOps yöntemini kullanarak, testler her gün sabah saat 6.00 ve akşam 18.00 da otomatik koşulmaktadırlar. Testinium için 93 koşum sayısı ve dolaysıyla 46 gün verisi Şekil 3'de verilmektedir.

Şekil 3'de gösterilen hata verileri ve eğilimlerinden anlaşıldığı üzere Testinium için hata oranı genel olarak daha azdır, ama TEY2 sistemi için çoğu günde yüksek hata oranı görülmektedir. TEY2 datasının bilgi güvenliği dolayısı ile bu gözlemin kök ve nedenlerini bu makalede detaylı olarak vermemekte ve tartışmamaktayız. Bununla birlikte Testinium i*le karşılaştırmalı olarak TEY2 datası incelendiğinde başarılı testlerin sayısının oranlarındaki fark net bir şekilde görülmektedir. Buda test süreçlerindeki iyileştirmenin arttırılmasına paralel olarak TEY' in kalitesinin de artmasına olanak sağlamaktadır.





Şekil 3. Gauge testlerin belirli bir zaman döneminde hata bulma trendi.

6 Sonuç ve gelecek çalışmalar

Bu keşifsel vaka-çalışmasında (*İnglizce:* exploratory case-study), *Gauge* adlı test aracı kullanılarak, yeni nesil otomasyon çerçevesinde yürütülebilir test spesifikasyonları geliştirerek, çalışmaların sonuçları bir deneyim raporu olarak sunuldu. Projede şu ana kadar, akademik yöntemleri kullanarak, başarılı bir şekilde test işlemleri yürütülmüştür. Keşifsel vaka-çalışmamız kapsamında, projemizde bir sonraki adım olarak belirli araştırılması gereken konular önümüzde mevcuttur: (1) geliştirilen testlerin hata bulma gücünü (*İngilizce:* fault detection effectiveness), kod ve gereksinimlerin kapsama oranını (coverage) ölçmek; (2) Test-edilen yazılım (TEY) bakım ve evrimi ile beraber test kodlarının bakımı [34].

Teşekkür: Bu çalışma, "TESTOMAT – The Next Level of Test Automation" başlıklı AB ITEA3 projesi ve Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (TÜBİTAK) 9180076 sayılı projesi tarafından desteklenmiştir.

Kaynakça

1. Britton, T., et al., *Reversible Debugging Software.* University of Cambridge, Judge Business School, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.370.9611, 2013.
2. Amannejad, Y., et al. *A search-based approach for cost-effective software test automation decision support and an industrial case study*. in *Proc. of International Workshop on Regression Testing, co-located with the IEEE International Conference on Software Testing, Verification, and Validation*. 2014.
3. Garousi, V. and F. Elberzhager, *Test automation: not just for test execution.* IEEE Software, 2017. **34**(2): p. 90-96.
4. Garousi, V. and M. Felderer, *Developing, verifying and maintaining high-quality automated test scripts.* IEEE Software, 2016. **33**(3): p. 68-75.
5. Infosys Co., *Case Study-Automating testing for a leading insurance company in Europe and the United States.* https://www.infosys.com/industries/insurance/case-studies/Pages/automated-testing.aspx, Last accessed: Apr. 2017.
6. Graham, D. and M. Fewster, *Experiences of Test Automation: Case Studies of Software Test Automation*. 2012: Addison-Wesley Professional; 1 edition.
7. Garousi, V., et al. *Türkiye'deki yazılım test uygulamaları anketi*. in *Ulusal Yazılım Mühendisliği Sempozyumu (UYMS)*. 2013.
8. Altıntaş, A.B., *Cucumber ve Java ile Davranışa yönelik geliştirme nasıl yapılır (BDD – Behavioral Driven Development) ?* https://kodcu.com/2016/07/cucumber-ve-java-ile-davranisa-yonelik-gelistirme-nasil-yapilir-behavioral-driven-development-bdd/, Last accessed: May 2019.
9. Maliackal, Z., *Why we built Gauge.* https://blog.getgauge.io/why-we-built-gauge-6e31bb4848cd, Last accessed: May 2019.
10. Matts, C., *The tragedy of Given-When-Then.* https://theitriskmanager.com/2019/04/06/the-tragedy-of-given-when-then/, Last accessed: May 2019.
11. Garousi, V., et al., *A survey of software engineering practices in Turkey.* Journal of Systems and Software, 2015. **108**: p. 148-177.
12. KIZILCA, H. and A.E. YILMAZ, *Doğal Dilde Yazılmış Gereksinimlerin Analiz Yöntemleri ve bu Yöntemlerin Türkçe için Uygulanabilirliği.* Yazılım Kalitesi ve Yazılım Geliştirme Araçları Sempozyumu, 2008: p. 69-77.
13. Garousi, V., S. Bauer, and M. Felderer, *NLP-assisted software testing: a systematic review.* arXiv preprint arXiv:1806.00696, 2018.
14. Raulamo, P., M.V. Mäntylä, and V. Garousi. *Choosing the right test automation tool: a grey literature review*. in *International Conference on Evaluation and Assessment in Software Engineering*. 2017. Karlskrona, Sweden.
15. Garousi, V., et al. *What industry wants from academia in software testing? Hearing practitioners' opinions*. in *International Conference on Evaluation and Assessment in Software Engineering*. 2017. Karlskrona, Sweden.
16. Garousi, V., K. Petersen, and B. Özkan, *Challenges and best practices in industry-academia collaborations in software engineering: a systematic literature review.* Information and Software Technology, 2016. **79**: p. 106–127.

17. Garousi, V. and K. Herkiloğlu. *Selecting the right topics for industry-academia collaborations in software testing: an experience report*. in *IEEE International Conference on Software Testing, Verification, and Validation*. 2016.
18. Rafi, D.M., et al. *Benefits and limitations of automated software testing: Systematic literature review and practitioner survey*. in *Proceedings of the International Workshop on Automation of Software Test*. 2012.
19. Ramler, R. and K. Wolfmaier. *Economic perspectives in test automation: balancing automated and manual testing with opportunity cost*. in *Proceedings of international workshop on Automation of software test*. 2006. ACM.
20. Garousi, V. and E. Yıldırım. *Introducing automated GUI testing and observing its benefits: an industrial case study in the context of law-practice management software*. in *Proceedings of IEEE Workshop on NEXt level of Test Automation (NEXTA)*. 2018.
21. Jolly, S.A., V. Garousi, and M.M. Eskandar. *Automated Unit Testing of a SCADA Control Software: An Industrial Case Study based on Action Research*. in *IEEE International Conference on Software Testing, Verification and Validation (ICST)*. 2012.
22. CKC Seminars, *www.testautomationday.com*. Last accessed: May 2019.
23. Li, N., A. Escalona, and T. Kamal. *Skyfire: Model-based testing with cucumber*. in *IEEE International Conference on Software Testing, Verification and Validation*. 2016.
24. Sivanandan, S. *Agile development cycle: Approach to design an effective Model Based Testing with Behaviour driven automation framework*. in *Annual International Conference on Advanced Computing and Communications*. 2014.
25. Urul, G., V. Garousi, and G. Urul. *Gerçek Zamanlı Gömülü Yazılımlar için Test Otomasyonu: Türkiye Endüstrisinden Bir Yaklaşım ve Deneyim Raporu*. in *Ulusal Yazılım Mühendisliği Sempozyumu (UYMS)*. 2014.
26. Garousi, V., et al., *Experience in automated testing of simulation software in the aviation industry.* IEEE Software, July/August 2019.
27. Kitchenham, B.A., T. Dyba, and M. Jorgensen. *Evidence-based software engineering*. in *Proceedings of the international conference on software engineering*. 2004.
28. Starreveld, A., *How to create an effective test automation strategy.* https://medium.com/@abstarreveld/considerations-for-an-effective-test-automation-strategy-a5bd027b3fa3, Last accessed: May 2019.
29. Gauge team, *Writing Specifications in Gauge.* https://docs.gauge.org/latest/writing-specifications.html, Last accessed: May 2019.
30. Ammann, P. and J. Offutt, *Introduction to Software Testing*. 1st ed. 2008: Cambridge University Press.
31. Meszaros, G., *xUnit Test Patterns*. 2007: Pearson Education.
32. Abrahms, J., *Selenium's Page Object Pattern: The Key to Maintainable Tests.* https://justin.abrah.ms/python/selenium-page-object-pattern--the-key-to-maintainable-tests.html, Last accessed: Apr. 2017.
33. Garousi, V. and B. Küçük, *Smells in software test code: A survey of knowledge in industry and academia.* Journal of Systems and Software, 2018. **138**: p. 52-81.
34. Shewchuk, Y. and V. Garousi. *Experience with Maintenance of a Functional GUI Test Suite using IBM Rational Functional Tester*. in *Proceedings of the International Conference on Software Engineering and Knowledge Engineering*. 2010.

Derin Öğrenme Modellerinde İzge Tabanlı Hata Konumlandırma

Mehmet Melih Arıcı[1][0000−0003−2502−0828], Hasan Ferit Enişer[1][0000−0002−3259−8794], and Alper Şen[1][0000−0002−5508−6484]

Boğaziçi Üniversitesi , Bilgisayar Mühendisliği Bölümü
{mehmet.arici,hasan.eniser,alper.sen}@boun.edu.tr

Özet. Yapay Sinir Ağları (YSA), otonom araçlardan tıbbi teşhislere kadar birçok güvenlik öncelikli uygulamada kullanılmaktadır. YSA'ların kapsamlıca test edilmesi, beklenmeyen YSA davranışlarını engellemek ve onların operasyonlarının güvenilirliğini sağlamak için kaçınılmazdır. Bu çalışmada, YSA davranışlarını Java gibi bir programlama dili ile ifade edebilmemiz, YSA'ların kapsamlı bir şekilde test edilmesinde yazılım mühendisliği araçlarından faydalanmamızı sağladı. Yazılımdaki hata konumlandırmasından esinlenerek sunduğumuz beyaz kutu YSA test yöntemi ile nöronların isabet izgesini oluşturup, yetersiz YSA performansından sorumlu olduğu düşünülen şüpheli nöronları tespit ettik. Ayrıca, orijinal girdileri şüpheli nöronların aktivasyonunu artıracak şekilde modifiye ederek, hasım girdiler elde edebildik.

Anahtar Kelimeler: Yapay Sinir Ağları · Hata Konumlandırma · Test Girdisi Yaratma

Spectrum Based Fault Localization for Deep Learning Models

Mehmet Melih Arıcı[1][0000−0003−2502−0828], Hasan Ferit Enişer[1][0000−0002−3259−8794], and Alper Şen[1][0000−0002−5508−6484]

Bogazici University , Department of Computer Engineering
{mehmet.arici,hasan.eniser,alper.sen}@boun.edu.tr

Abstract. Artificial Neural Networks (ANNs) are increasingly deployed in safety-critical applications including autonomous vehicles and medical diagnostics. To avoid unexpected ANN behaviour and provide evidence for their trustworthy operation, ANNs should be thoroughly tested. In this study, representing ANNs with a programming language such as Java, enabled us to utilize software engineering tools for comprehensive testing of ANNs. Inspired by fault localization in software, the whitebox ANN testing approach that we presented in our paper, establishes the hit spectrum of neurons and identify suspicious neurons which are responsible for inadequate ANN performance. Moreover, we also modified original inputs to synthesize adversarial inputs that increase activation values of suspicious neurons.

Keywords: Deep Neural Networks · Fault Localization · Test Input Generation.

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

1 Giriş

Yapay Sinir Ağları (YSA), imge sınıflandırma, doğal dil işleme ve ses tanıma gibi birçok zorlu makine öğrenmesi görevlerinde, insan seviyesinde yetenekler göstermektedir. Bu etkileyici başarılar YSA'ları, özellikle hayati tehlike arz eden gerçek dünya uygulamalarında kullanma konusunda beklentileri yükseltmiştir.

YSA'ların yaygın olarak benimsenmesiyle beklenen faydalara rağmen, onların güvenlik öncelikli sistemlerde kullanımı, yüksek güvenilirlik derecesiyle nitelendirilmelidir. Güvenlik öncelikli alanlarda YSA'ların beklenen bir davranıştan veya doğru bir işlemden sapması durumunda insan hayatı tehlikeye girebilir veya ciddi finansal kayıplar ortaya çıkabilir. Bu sebeple YSA tabanlı sistemlerin, sadece yüksek seviyelerde güvenilirlik içerdikten sonra serbest kullanımına izin verilmelidir [1].

Yazılım testi, yazılım sisteminin kalitesini analiz etmek ve değerlendirmek için fiili bir araçtır [6]. Test etme, hem erken davranarak problemleri (hataları) bulur ve bunları ortadan kaldırarak riski azaltır, hem de test sonuçlarını kullanarak, sistemin gereken güvenlik seviyesine ulaşıp ulaşmadığını kanıtlar.

YSA testi alanındaki son çalışmalarda, YSA'lar için ilginç beyaz kutu ve siyah kutu test teknikleri önerilmiştir [4, 8, 12, 13, 17, 19]. Bazı çalışmalar mutasyon tabanlı sezgileri kullanarak eğitim verisini, hasım (adversarial) veriye çevirmiş [25]; simgesel yürütme [3], birleşimsel [13] ya da concolic [19] test uygulamışken, diğerleri nöron kapsamı [17] ve türevlerini [11], Değiştirilmiş Koşul / Karar Kapsamından (MC/DC) esinlenen bir kriter [18] gibi YSA'lara özgü kapsama kriterlerini ileri sürmüştür. Bu son ilerlemeler, geneksel yazılım test tekniklerinin YSA testine direk olarak uygunlanamasa bile, bu tekniklerin arkasında yatan karmaşık konseptlerin eğer doğru uygulanırsa, makine öğrenmesi alanında kullanışlı olabileceğini kanıtlamıştır. Bununla beraber, önerilen tekniklerin hiçbiri, hatalı davranışlardan en çok sorumlu olan sistem bölgelerini tespit eden *hata konumlandırmayı* [24] kullanmamaktadır.

Bu makalede, ileri beslemeli bir YSA'yı, bir programlama dili ile ifade eden, YSA'nın kaynak kodunu oluşturduktan sonra bir yazılım mühendisliği aracı ile analizini yapan, YSA'nın hatalı davranışında daha büyük pay sahibi olan *şüpheli nöronları* tespit eden ve doğru sınıflandırılmış girdileri kullanarak, tespit edilen şüpheli nöronları harekete geçiren yeni girdiler sentezleyebilen çalışmamızı sunacağız. Kusurlu yazılımı girdi olarak alıp, çıktı olarak şüpheli kod bölgelerinin sıralanmış listesini veren geleneksel hata konumlandırmaya [24] benzer şekilde, yöntemimiz, eğitilmiş bir YSA'nın nöronlarının davranışlarını analiz ederek onların isabet izgelerine ulaşır ve şüphelilik ölçütlerinden faydalanarak şüpheli nöronları tespit eder. Şüphelilik güdümlü test girdisi üretme algoritmamız, doğru sınıflandırılmış girdileri modifiye ederek şüpheli nöronlar için yüksek aktivasyon değerlerine ulaşan yeni girdiler sentezler. Popüler ve genel kullanıma açık MNIST [10] veri kümesi üzerinde yaptığımız ampirik değerlendirme, yeteriz ağ performansından sorumlu olan şüpheli nöronların yöntemimiz tarafından tespit edilebildiğini kanıtlamıştır. Ayrıca yöntemimiz, tespit edilen şüpheli nöronların aktivasyon değerlerini arttıran yeni girdiler sentezleyebilir. Bu girdiler çoğunlukla hasım girdiler olmaktadır.

Genel olarak, bu makaledeki katkılarımız şöyle sıralanabilir:

- İleri beslemeli YSA'ları, Java kaynak koduna çeviren bir yöntem önerdik.
- Üretilen kod üzerinde bir kod kapsamı aracı çalıştırarak YSA'larda hata konumlandırma tekniği sunduk.
- Konumlandırılan hatalı YSA elemanlarını (nöronlar) yüksek aktivasyon değerleri ile çalıştıran yeni girdiler sentezledik.
- Yöntemimizin geçerliliğini ve etkisini, genel kullanıma açık MNIST veri kümesi üzerinde kapsamlı bir şekilde değerlendirdik.

2 İlgili Çalışmalar

YSA'ların Test Edilmesi ve Doğrulanması. YSA'ların sürücüsüz vasıtalar, insansız hava araçları ve medikal resim analizi gibi bir çok güvenlik öncelikli sistemde kullanılmaya başlanması ve bunun yanında YSA'ların hasım girdilere karşı zayıf olduğunun gösterilmesi [20] araştırmacıları yazılım mühendisliği alanında geliştirilmiş olan yazılım test ve doğrulama tekniklerini YSA'ların güvenilirliğini sağlamak için kullanmaya itmiştir [8, 11, 17]. Literatürde, bunlara örnek olarak DeepXplore [17] yazılımdaki kapsama (ya da test yeterliliği) kriterlerine benzer nöron kapsama kriteri önerilmiştir. Bunun yanında, yazarlar, yazılımdaki ayrımsal test yöntemini öğrenme algoritmalarının bir kısıtı olan test kahini probleminin aşılması için sunmuştur. DeepGauge [11] nöron kapsaması kriterini geliştirerek yeni kapsam kriterleri sunmuştur. Bunların dışında yazılımdaki metamorfik test [21], mutasyon testi [12] ve concolic test [19] yöntemlerini YSA'lara uyarlayan çalışmalar bulunmaktadır.

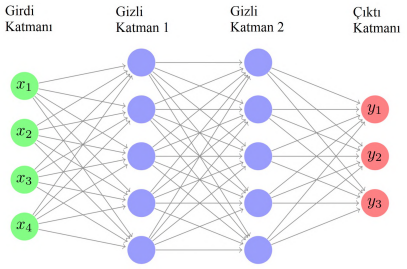
Matematiksel YSA doğrulaması güvenilir YSA kararları için bir garanti sağlama gayesi gütmektedir [4]. Soyutlama arıtma (abstraction refinement) yöntemi Pulina ve ark. tarafından Sigmoid aktivasyonu kullanılan küçük ölçekli YSA'ları doğrulamak için kullanılmıştır. AI^2 [2] ise doğrulama için soyut yorumlama (abstract interpretation) yöntemini kullanır ve daha büyük ölçekli ağlarda çalışabilir. Reluplex [7] SMT tabanlı bir yöntem önererek ReLU aktivasyonlu YSA'ların üzerinde belirli özelliklerin doğrulanmasını sağlar. Son olarak DLV [22] bölgesel YSA sağlamlığını doğrulamak için formel bir yöntem önerir.

Bizim bu çalışmada sunduğumuz YSA'lar için hata konumlandırma yöntemi, literatürdeki çalışmalara dikey olarak konumlanmaktadır.

Yazılımda Hata Konumlandırma. Hata konumlandırma, yazılım hata ayıklama ve program onarma gibi birçok yazılım mühendisliği alanında yaygın olarak çalışılmış bir konudur. Hata konumlandırma araştırmalarının odak noktası, mühendislik eforlarını düşürerek [16], yazılım hatalarının temel nedenini izole eden, tespit yöntemlerinin ve şüphelilik ölçütlerinin geliştirilmesidir. Bu çalışmada izge tabanlı hata konumlandırma yöntemlerinden faydalanılmıştır. Önde gelen izge tabanlı hata konumlandırma yöntemlerinden bazıları şu çalışmalarda bulunabilir [5, 9, 23].

3 Temel Bilgiler

3.1 Yapay Sinir Ağları

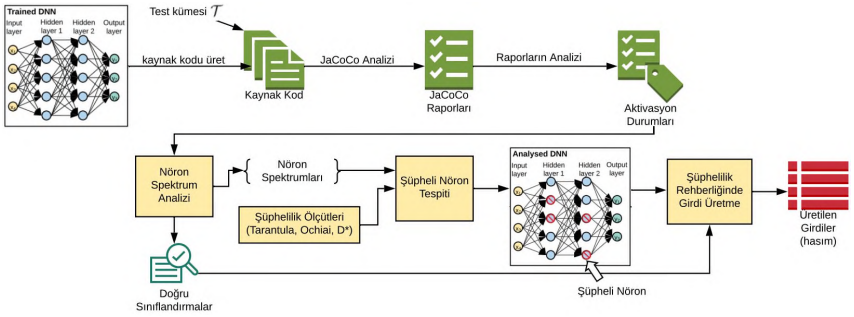


Şekil 1. Dört katmandan oluşan, tam bağlantılı, ileri beslemeli bir yapay sinir ağı.

YSA'lar güçlü hesaplama kabiliyetleriyle, son dönemlerin en ilgi duyulan konulardan birisi haline gelmiştir. YSA'ların birçok çeşidi olmakla birlikte, en temel olanı ileri-beslemeli yapay sinir ağlarıdır. İleri-beslemeli YSA, birbirine bağlı nöronlardan oluşmaktadır. Derin ileri-beslemeli bir YSA'da bir tane girdi katmanı, bir tane çıktı katmanı ve en az bir tane gizli katman bulunması gerekir. Her katman, bir dizi nörona sahiptir. Her bir nöron çıktısı, bir sonraki katmandaki nöronlara gitmeden önce, doğrusal olmayan aktivasyon fonksiyonlarından geçer. YSA'lar için en bilinen aktivasyon fonksiyonları; sigmoid, hyperbolic tangent (tanh), ReLU (Rectified Linear Unit) ve Leaky ReLU'dur. ReLU günüzmüzde, diğer aktivasyon fonksiyonlarına göre daha fazla kullanılmaktadır. ReLU aktivasyon fonksiyonu $f(x) = max(0, x)$, nöron çıktısı ve sıfır değeri arasından maksimum olanı seçer. ReLU aktivasyon fonksiyonunun modifiye edilmiş versiyonu olan Leaky ReLU $f(x) = max(\alpha * x, x)$, negatif çıktıları çok küçük bir değer (Örn. $\alpha = 0.01$) ile çarpar. Girdi katmanı hariç, diğer katmanlardaki nöronlar arasında bağlantılar vardır ve her bağlantının bir ağırlığı vardır. YSA'ların nöronlar arasındaki ağırlıkları, eğitim (training) adı verilen periyot süresince, geri yayılım (back propagation) adı verilen bir algoritma ile belirlenmektedir.

3.2 Yazılım Hata Konumlandırması

Yazılım Hata Konumlandırması (YHK), hata içerme olasılığı yüksek olan kaynak kodu elemanlarını (Örn. komutlar,bildirimler) tespit etmeye yarayan bir beyaz kutu test tekniğidir. Geleneksel yazılımlar için YHK süreci, girdi olarak, test edilen sisteme karşılık gelen P programını ve $\mathcal{T}$ test kümesini alır. Daha sonra, P'yi, $\mathcal{T}$'ye karşı test eden hata konumlandırması (HK) tekniğinden yararlanarak, başarılı olan ve hata veren testleri temsil eden alt kümeleri belirler. Bu alt kümeleri ve $p \in P$ program elemanlarını kullanan HK tekniği, hata tespit verisini bulur ve bu veri daha sonra HK tarafından, her bir p program elamanının şüpheliliğini ölçmek için kullanılır. En çok kullanılan HK tekniği olan izge-tabanlı HK, başarılı ve başarısız test işlemlerinin program izlerini kullanarak, p program elemanı için (e_s, e_f, n_s, n_f) listesini oluşturur. e_s ve e_f (n_s ve n_f) değerleri, verilen test edilmiş (veya edilmemiş) program elamanının, sırasıyla kaç defa başarılı ve başarısız olduğunu ifade eder. İzge-tabanlı HK ölçütü, oluşturulan listeyi alarak, program elemanlarını, azalan şüphelilik derecelerine göre sıralar. Böylece yazılım mühendisleri, program elemanlarını kontrol edebilir ve hataları etkili bir şekilde bulabilir.



Şekil 2. Akış Şeması

4 Yöntem

Bu kısımda, yapay sinir ağlarının, Java programlama dili ile ifade edilmesini, bir yazılım mühendisliği aracı kullanarak, elde edilen kod üzerinden şüpheli nöronların nasıl tespit edilebileceğini ve tespit edilen şüpheli nöronlar vasıtasıyla orijinallerine çok yakın hasım girdi elde etmeyi gösteren bir yöntem sunacağız.

Şekil 2, yöntemimizin akış şemasını göstermektedir. İlk olarak, daha önceden eğitilmiş olan yapay sinir ağı, geliştirdiğimiz algoritma ile Java kaynak koduna çevrilmiştir. Bu yöntem ile elde edilmiş kaynak kod üzerinden bir yazılım mühendisliği aracı kullanılarak, yapay sinir ağına verilen bir girdi için, her bir nöronun aktive olup olmadığını tespit edilmiştir. Elde ettiğimiz nöron aktivasyon bilgileri, daha sonra, verilen yapay sinir ağının şüpheli nöronlarını tespit etmemizi sağlayan yazılım mühendisliği ölçütlerinin hesaplanmasında kullanıldı. Şüpheli nöronları tespit etmemiz, daha önce doğru sınıflandırılmış olan girdileri, şüpheli nöronların aktivasyon değerlerini artıracak şekilde modifiye edip, yapay sinir ağını kandırabilen hasım girdiler elde etmemizi sağladı.

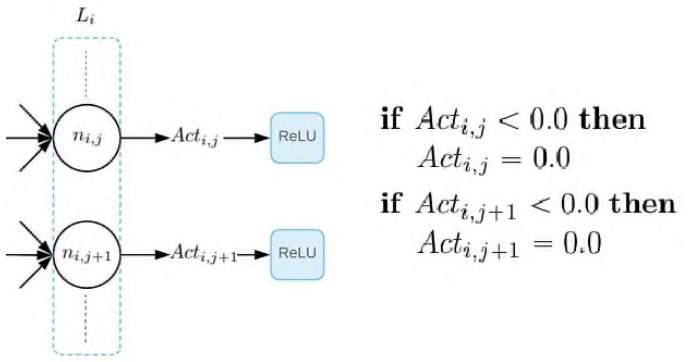
Yöntemimizi açıklamak için kullanacağımız notasyonlar şu şekildedir: $\mathcal{N}$, l tane katmana sahip bir YSA'dır. Her bir katman L_i, s_i tane nöron içermektedir. $(0 <= i < l)$ Nöron n_{ij}, katman L_i'deki j'ninci sıradaki nöronu temsil etmektedir. W_i, $(0 <= i < l)$, YSA $\mathcal{N}$'nin L_i katmanındaki ağırlıkları ifade etmektedir. YSA $\mathcal{N}$'nin her katmanı için, nöron aktivasyon değerlerini tutan aktivasyon listeleri vardır. Örneğin Act_1 ilk katmandaki, Act_l ise en son katmana ait nöron aktivasyon değerlerini tutmaktadır. Herhangi bir katmana ait aktivasyon listesinden, o katmandaki herhangi bir nöronun aktivasyon değerine ulaşılabilir. Örneğin, ikinci katmandaki onuncu nöronun aktivasyon değeri $Act_{2,10}$ elemanına bakılarak bulunabilir.

4.1 Yapay Sinir Ağının Java Programlama Dili İle İfade Edilmesi

Bu kısımda, yapay sinir ağlarını, nöronların aktive olma durumunu kontrol edebileceğimiz bir şekilde, Java programlama dili ile nasıl ifade ettiğimizi göstereceğiz. Temel olarak bir nöronu, aktivasyon değeri belirli bir sınırın üzerinde ise aktif, değilse inaktif olarak kabul ediyoruz. Her bir nöron için iki dalı olan bir dallanma mekanizması kullanmak ve dallanma kapsamı analizi (branch coverage) yapabilen bir yazılım mühendisliği aracını bu sistem üzerinde çalıştırmamız sayesinde,

verilen bir girdi için yapay sinir ağındaki tüm nöronların aktivasyon durumuna ulaşabildik.

Yapay sinir ağlarında en çok kullanılan aktivasyon fonksiyonlarından ReLU (Rectified Linear Unit) ve Leaky ReLU, bize istediğimiz iki dalı olan dallanma mekanizmasını sağlayabildi. Bu aktivasyon fonksiyonları, koşullu yapılarından dolayı, dallanma talimatları (instruction) olarak yazılabilir. Bu durumda ReLU'ya karşılık gelen talimat, $if(x > 0)$ *then return* x *else return* 0, ve Leaky ReLU'ya karşılık gelen talimat ise, $if(x > 0)$ *then return* x *else return* $\alpha * x$, olacaktır. Şekil 3'de bir YSA'daki iki nöron ve onların aktivasyon fonksiyonlarına karşılık gelen örnek bir kod bloğu gösterilmiştir.



Şekil 3. Katman L_i'ye ait iki nöron; $n_{i,j}$ ve $n_{i,j+1}$ (solda). Bu iki nöronun ReLU aktivasyon fonksiyonlarını ifade eden kod bloğu (sağda).

Bir katmana ait sıralı if dizisi, o katmandaki son nöronun aktivasyon fonksiyonunu ifade eden if ile biter. Bir sonraki katmandaki (L_{i+1}) nöron aktivasyon fonksiyonlarını ifade eden if dizisine geçmeden önce, bu katmanın aktivasyon listesi olan Act_{i+1}'in oluşturulması gerekmektedir. Act_{i+1}'in başlangıç değerleri, Act_i ile W_{i+1}'in skaler çarpımı ile bulunur. Bu durumda katman L_{i+1}'in aktivasyon listesi, $Act_{i+1} = Act_i \cdot W_{i+1}$, olarak başlatılmış olur. Act_{i+1}'i başlattıktan sonra, L_{i+1}'e ait if dizisini oluşturabiliriz.

JaCoCo, Java programlama dili için geliştirilen ve satır, dallanma kapsamı gibi ölçütleri analiz etmemizi sağlayan açık kaynak kodlu bir yazılım projesidir. JaCoCo, verilen kod üzerinde analiz yaptıktan sonra kod kapsamı raporları üretebilmektedir. JaCoCo'yu, üretilen yapay sinir ağı kodu üzerinde çalıştırdık. Üretilen raporu otomatik olarak analiz edebilen bir betik ile dallanma kapsamı bilgisine ulaştık. Bu betik ile, nöron aktivasyon fonksiyonlarını ifade eden her bir if deyiminin kapsanıp kapsanmadığına bakıyoruz. Eğer bir if deyimi kapsanmışsa, bunun anlamı, ilgili nöronun aktivasyon değerinin sıfırdan küçük olmasıdır ve dolayısıyla bu nöron inaktifdir. Öte yandan, eğer bu if deyimi kapsanmamışsa, bu nöron aktiftir. Bu kısımda nöron aktivasyon durumlarını tespit etmemiz, bir sonraki kısımda şüpheli nöronları tespit etmemize yardımcı oldu.

Tablo 1. Şüphelilik Ölçütleri

Şüphelilik Ölçütü	**Cebirsel Formül**
Tarantula [5]:	$\frac{attr_n^{\text{af}}/(attr_n^{\text{af}}+attr_n^{\text{nf}})}{attr_n^{\text{af}}/(attr_n^{\text{af}}+attr_n^{\text{nf}})+attr_n^{\text{as}}/(attr_n^{\text{as}}+attr_n^{\text{ns}})}$
Ochiai [15]:	$\frac{attr_n^{\text{af}}}{\sqrt{(attr_n^{\text{af}}+attr_n^{\text{nf}})\cdot(attr_n^{\text{af}}+attr_n^{\text{as}})}}$
D* [23]:	$\frac{attr_n^{\text{af}*}}{attr_n^{\text{as}}+attr_n^{\text{nf}}}$

$* > 0$ bir değişkendir. En yaygın değerlerden biri olarak $* = 3$ kullandık. [16, 24].

4.2 Şüpheli Nöronların Tespit Edilmesi

Bu kısımda, bir önceki kısımda elde edilen nöron aktivasyon durumlarından da faydalanılarak, YSA'nın yetersiz performans göstermesine neden olan şüpheli nöronlar tespit edilmiştir. Şüpheli nöronların tespiti, her bir nöron için şüphelilik skoru hesaplayan bir şüphelilik ölçütü kullanılarak yapılmıştır. Bir nöron için şüphelilik skorunun yüksek çıkması, o nöronun tatmin edici bir şekilde eğitilmediğinin bir göstergesidir ve bu nörona ait ağırlıkların tekrar ayarlanmaya ihtiyaç duyduğunu ifade eder.

Şüpheli nöronların tespiti, bir nöronun işlem modelini yakalayan niteliklerin oluşturulmasıyla başlar. Bu nitelikler şu şekilde tanımlanmıştır: $attr_n^{\text{as}}$ ve $attr_n^{\text{af}}$ sırasıyla, nöron n'nin kaç defa aktif olduğunu (aktivasyon fonksiyonunun çıktısının, $\phi(t, n)$, belirlenmiş bir sınır değerinden yüksek olması) ve YSA'nın kaç defa başarılı ve hatalı karar verdiğini ifade eder. Benzer şekilde, $attr_n^{\text{ns}}$ ve $attr_n^{\text{nf}}$, nöron n'nin aktif olmadığı durumları kapsar. YSA'nın gizli katmanlarındaki nöronların davranışları, belirli bir $\mathcal{T}$ test kümesi için, analiz edilerek, her nöron için İsabet İzgesi (II) bir araya getirilir. İsabet izgesi her bir nöronun verilen test kümesindeki, aktivasyon değerleri açısından davranışını özetleyen bir listedir.

Tanım 1. *$\mathcal{N}$ bir YSA ve $\mathcal{T}$ bir test kümesi olmak üzere, herhangi bir nöron n'nin ($n \in \mathcal{N}_{\mathcal{H}}$) isabet izgesini gösteren liste şu şekilde ifade edilir: $II_n = (attr_n^{as}, attr_n^{af}, attr_n^{ns}, attr_n^{nf})$.*

Her nöronun II değerlerinin toplamı $\mathcal{T}$ test kümesinin büyüklüğüne eşit olmalıdır.

Tanım 2. *II_n, verilen bir nöron n'nin ($n \in \mathcal{N}_{\mathcal{H}}$) isabet izgesi olmak üzere, bir nöronun izge-tabanlı şüpheliliği* ŞüpheHesapla$_n : II_n \rightarrow \mathbb{R}$ *fonksiyonu ile tespit edilir.*

Bu makalede üç farklı şüphelilik ölçütü kullandık. (Tarantula [5], Ochiai [15] ve D* [23]) Bu ölçütlerin tanımları Tablo 1'de verilmiştir. Bu şüphelilik olçütlerinin altında yatan prensip, YSA'nın yanlış karar verdiği test girdileri için bir nöron ne kadar sık aktive edilmişse, bu nöron, YSA'nın doğru karar verdiği test girdileri için daha az sıklıkta aktif olur ve daha şüphelidir.

Algoritma 1 genel hatlarıyla, YSA'daki k tane en şüpheli nöronu tespit etmeye ve seçmeye yarayan adımları göstermektedir. İlk olarak, bahsi geçen YSA'dan Java kaynak kodu üreten KaynakKoduÜret fonksiyonu çağrılır. JacocoAnalizi fonksiyonu, üretilen kaynak kodunu ve test kümesini kullanarak,

Algorithm 1 Şüpheli Nöronların Tespit Edilmesi

Girdiler:
$\mathcal{N}$: Yapay Sinir Ağı, $\mathcal{T}$: Test Kümesi , k: Şüpheli Nöron Sayısı
Çıktılar:
SN: Şüpheli Nöron Listesi

$S \leftarrow \varnothing$ ▷ şüphelilik vektörü
$\mathcal{N}' =$KaynakKoduÜret($\mathcal{N}$) ▷ YSA $\mathcal{N}$'den kaynak kod üret
$A^{\mathcal{T}} =$JacocoAnalizi($\mathcal{T}, \mathcal{N}'$) ▷ tüm nöronların aktivasyonlarını hesapla
Her bir $n \in \mathcal{N}$ **için**:
  $II_n \leftarrow \varnothing$ ▷ nöron n'nin isabet izge vektörü
  Her bir $p \in \{as, af, ns, nf\}$ **için**:
    $a_n^p =$NitelikHesapla($A_n^{\mathcal{T}}, p$) ▷ nöron n'nin p niteliğini hesapla
    $II_n \leftarrow II_n \cup \{a_n^p\}$ ▷ niteliği İsabet İzgesi'ne (bkz. Tan. 1) ekle
  $S \leftarrow S \cup \{$ŞüpheHesapla(II_n)$\}$ ▷ şüphelilik skorunu hesapla (bkz. Tan. 2)
SN $\leftarrow \{n |$ŞüpheHesapla(II_n) $\in$ Seç(S, k)$\}$ ▷ en şüpheli k tane nöronu seç

YSA'daki *her bir nöronun*, tüm test girdisi ($\mathcal{T}$) üzerindeki aktivasyon bilgisini tutan $A^{\mathcal{T}}$ dizisini hesaplar. Ardından YSA'daki her nöronun isabet izgesini ifade eden listenin oluşturulması gerekmektedir. Bu amaçla NitelikHesapla fonksiyonu; her nöron n ve bu nöronun her bir p niteliği için, bu nörona ait aktivasyon bilgisi $A_n^{\mathcal{T}}$'yı da kullanarak, nitelik p'nin değerini hesaplar. Nöron n'ye ait tüm nitelik değerlerinin hesaplanmasıyla, bu nörona ait isabet izgesi II_n oluşturulmuş olur. ŞüpheHesapla fonksiyonu, oluşturulan isabet izgesini ve Tarantula vb. şüphelilik ölçütlerini kullanarak, bu nöronun şüphelilik skorunu hesaplar. Her bir nöron için elde edilen bu şüphelilik skoru, algoritmanın sonunda, k tane en şüpheli nöronu seçmek için kullanılacaktır. Eğer birden çok nöron aynı şüphelilik skorunu elde ederse, algoritma derin katmanlardaki (çıktı katmanına yakın olan katmanlar) nöronlara öncelik verir. Bunun nedeni, derin katmanlardaki nöronların, girdi uzayının daha mantıklı temsillerini öğrenebilmesidir.

4.3 Şüpheliliğe Dayalı Girdi Sentezi

Bu kısımda, en şüpheli k tane nöronun seçilmesinden sonra, bu nöronları aktive eden hasım girdi sentezlenmesini göstereceğiz. Girdi sentezlenmesinin altında yatan neden, şüpheli nöronların aktivasyon değerlerinin arttırılmasıyla, bu nöronlar tarafından tüm sinir ağı boyunca hesaplanan bozulmuş bilgilerin yayılması ve böylece YSA'nın karar sınırının kaymasına neden olmaktır. Buna ulaşmak için, YSA'nın doğru sınıflandırma yaptığı test kümesinden alınan test girdilerine, hedefli modifikasyon yapılır ve böylece YSA'nın kararını farklı bir bölgeye çekmek amaçlanır.

Şüpheli nöronların aktivasyon değerlerini arttırmak için doğru sınıflandırılmış girdilerin ne kadar değiştirileceğini belirlemeyi amaçlayan girdi sentezleme süreci, gradyan çıkışı algoritması ile desteklenir. YSA tarafından doğru sınıflandırılmış her bir test girdisi $t \in T_s$ için, şüpheli nöronların değerleri ve gradyanları alınır. Bu gradyan değerleri t girdisine uygulanarak yeni girdi elde edilir. Fakat yeni girdinin orijinal girdiden çok farklı olmaması için belli kısıtlar konulur.

Örneğin, bir imge sınıflandırma veri kümesi için bu kısıtlama, sentezlenen imgelerin piksel değerlerini belirli bir aralıkta tutmak olabilir. (Örn. MNIST veri kümesi için $0 - 1$ arasında)

Değerlendirme kısmında, şüphelilik ölçütlerinin, YSA'nın daha önce doğru sınıflandırdığı girdileri yanlış sınıflandırmasına yol açan hasım girdileri üretibildiğini göreceğiz. Bu durum, tespit edilen şüpheli nöronların, yetersiz YSA performansında büyük rol oynadığını gösterir.

5 Değerlendirme

Bu kısımda, bu makalede sunulan tekniklerin, yapay sinir ağları için izge tabanlı hata konumlandırmada ne kadar başarılı olduğunu gösterdik. Deneylerimizde kullandığımız MNIST, 60,000 eğitim ve 10,000 test örneğine sahip el yazısı rakam veri kümesidir. Her girdi 28x28 pikselli imgedir ve sınıf etiketi 0-9 arasındadır.

Deneylerimizi sırasıyla, 5 katmana (her katmanda 30 nöron), 6 katmana (her katmanda 25 nöron) ve 8 katmana (her katmanda 20 nöron) sahip üç farklı YSA ile yaptık. Her YSA, farklı mimari ve eğitilebilir parametre sayısına sahip olmakla beraber, tam bağlantılı katmanlardan oluşmaktadır. Her YSA, aktivasyon fonksiyonu olarak Leaky ReLU [14] kullanır ($\alpha = 0.01$). Fakat yöntemimiz diğer aktivasyon fonksiyonları ile de çalışabilmektedir.

Değerlendirmemizi, Tarantula [5], Ochiai [15] ve D* [23] şüphelilik ölçütleri üzerinde yaptık ve yöntemimizin etkisini, farklı sayıda şüpheli nöronlar ($k \in \{1, 2, 3, 5, 10\}$) kullanarak gözlemledik. 4.3 kısmında belirtilen gradyan değerlerini, gerçekçi girdiler sentezlemek için, orijinal girdinin maksimum piksel değerinin en fazla %10'u (L_∞) olacak şekilde kısıtladık. Tablo 3'te gösterildiği gibi sentezlenen girdiler, orijinal girdilere çok benzerdir.

5.1 Araştırma Soruları

Aşağıdaki araştırma sorularını, deneylerimiz ile cevaplandırmaya çalışacağız.

AS1 (Onaylama): Tekniğimiz, şüpheli nöronları etkili bir şekilde bulabiliyor mu?

AS2 (Karşılaştırma): Farklı şüphelilik ölçütleri şüpheli nöronları bulmada ne kadar etkilidir?

AS3 (Benzerlik): Sentezlenen girdiler ne kadar gerçekçidir?

5.2 Araştırma Sonuçları

AS1 (Onaylama). Tekniğimizi, değerlendirmemizde kullandığımız 3 farklı YSA'ya uyguladık. Şüphelilik ölçütlerimizden birini seçerek, verilen bir test kümesi $\mathcal{T}$ için, daha önceden eğitilmiş olan YSA'yı Java kodunu çevirerek, JaCoCo aracı ile analiz ettik ve en yüksek şüphelilik skoruna sahip k tane nöronu belirleyerek, doğru sınıflandırılmış girdilerden, şüpheli nöronların aktivasyonlarını arttıran yeni girdiler sentezledik. Daha sonra, standart performans ölçütlerinden olan çapraz-entropi kaybını (cross-entropy loss) (çıktı ve hedef dağılımları arasındaki fark) ve doğruluk skorunu (tüm girdiler içinde doğru sınıflandırılanların yüzdesi) kullanarak, YSA'nın sentezlenen girdiler üzerindeki tahmin performansını ölçtük. YSA analizinin her sınıfa özel olduğu unutulmamalıdır. Bunun nedeni, aynı sınıfa ait girdilerin aktivasyon modellerinin birbirleriyle benzer olmasıdır.

Tablo 2, yöntemimizin MNIST veri kümesinde, Tarantula (T), Ochiai (O), DStar (D) ve rastegele (random) seçme stratejisi kullanarak, farklı sayılarda (k) şüpheli nöronlar için, ortalama kayıp ve doğruluk miktarlarını göstermektedir. Tablo 2'teki her bir değer, rastgele R hariç, 100 tane sentezlenmiş girdi (sınıf başına 10 tane) üzerinden ortalama alınarak bulunmuştur. R için, 5 farklı denemede, 500 tane sentezlenmiş girdi (sınıf başına 50 tane) toplanarak, bulgularımızın şans yardımıyla elde edilmiş olma riski düşürülmüştür.

Tablo 2. MNIST veri kümesinden sentezlenen girdilerin doğruluk ve kayıp değerleri. Şüphelilik ölçütü başına en iyi sonuçlar, kalın rakamlarla gösterilmiştir. (k:#şüpheli nöronlar, T:Tarantula, O:Ochiai, D:D*, R:Rastgele)

k	**Ölçüt**	**MNIST_1**				**MNIST_2**				**MNIST_3**			
		T	O	D	R	T	O	D	R	T	O	D	R
1	Kayıp	3.55	**6.19**	4.03	2.42	3.48	3.53	**3.97**	2.78	7.35	**8.23**	6.36	3.66
	Doğruluk	0.26	**0.16**	0.2	0.59	0.3	**0.2**	0.5	0.49	0.16	**0.1**	0.13	0.39
2	Kayıp	3.73	**6.08**	3.18	2.67	3.12	3.76	**4.08**	0.9	4.27	**6.81**	6.5	3.06
	Doğruluk	**0.16**	0.23	0.4	0.58	0.23	0.23	**0.13**	0.77	0.29	**0.13**	0.26	0.56
3	Kayıp	4.1	6.19	**6.25**	1.14	2.39	**3.94**	3.04	1.61	3.33	**7.59**	6.98	2.91
	Doğruluk	**0.23**	**0.23**	0.33	0.77	0.46	0.26	**0.23**	0.67	0.26	**0.06**	0.16	0.61
5	Kayıp	4.63	6.68	**6.97**	1.1	2.49	**3.64**	3.48	0.94	4.15	**7.22**	6.47	1.22
	Doğruluk	0.23	0.23	**0.13**	0.79	0.26	0.26	**0.2**	0.73	0.16	**0.1**	0.26	0.77
10	Kayıp	4.97	6.95	**7.4**	1.3	2.08	3.06	**3.82**	0.49	4.45	**7.16**	5.9	0.57
	Doğruluk	0.23	**0.2**	0.23	0.75	0.4	**0.23**	0.26	0.86	**0.13**	**0.13**	**0.13**	0.87

Beklenildiği gibi, herhangi bir şüphelilik ölçütü (T, O, D) ile çalışan yöntemimiz, R'ye kıyasla, çok daha düşük tahmin performansı vermiştir. Bu sonuçlar gösteriyor ki, tespit edilen k tane nöron, gerçekten şüphelidir ve daha önce doğru sınıflandırılan girdileri, bu nöronların aktivasyon değerlerini arttıracak şekilde modifiye etmek, onları hasım girdilerine çevirebilmektedir.

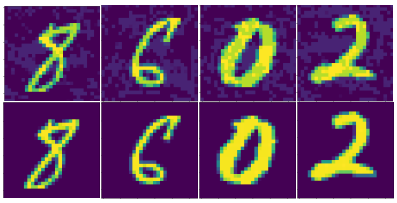
AS2 (Karşılaştırma). Yöntemimizde kullandığımız farklı şüphelilik ölçütleri için, Mann-Whitney testinden yararlanarak, ikili karşılaştırmalar yaptık. Ochiai metriğinin farklı k değerleri için, diğerlerine kıyasla, daha iyi sonuçlar elde ettiğini gözlemledik. Bu sonuç bize gösteriyor ki, Ochiai tarafından belirlenen şüpheli nöronlar, YSA'nın yetersiz performansında, daha fazla pay sahibidir. Bulgularımız, çeşitli şüphelilik ölçütlerinin, yöntemimizde kullanılabileceğini ve başarılı performanslar gösterebileceğini gösterdi.

AS3 (Benzerlik). Yöntemimizin gerçekçi girdi üretme becerisini gözlemleyebilmek için, doğru sınıflandırışmış, orijinal girdiler ile sentezlenen girdiler arasındaki uzaklığı inceledik. Orijinal ve sentezlenen girdiler arasındaki uzaklık, çeşitli uzaklık ölçütleri (L_1 Manhattan, L_2 Euclidean, $L\infty$ Chebyshev) kullanılarak hesaplanmıştır. Tablo 3'te farklı şüphelilik ölçütleri için, orijinal girdiler ile sentezlenen girdiler arasındaki uzaklığa baktık. T, O ya da D* kullanılarak sentezlenen girdiler, rastgele seçim stratejisi (L_1 uzaklığı) ile elde edilen girdilere çok yakındır. [0,1] aralığı arasından değer olan 784 tane pikselden oluşan bir MNIST girdisi için, girdi başına ortalama sarsım miktarı, toplam mümkün olsan sarsım miktarının %5.28'inden küçüktür (L_1 uzaklığı). Dolayısıyla, sentezlenen girdiler ile onların orijinal versiyonları arasındaki farklılık çok azdır. Bu sonuçları düşünerek,

yöntemimizin, orijinallerine çok benzeyen hasım girdiler üretme konusunda başarılı olduğunu söyleyebiliriz.

Tablo 3. Sentezlenen girdiler ile orijinal girdiler arasındaki uzaklık.

	Uzaklık Ölç.		
Şüphelilik Ölçütü	L_1	L_2	L_∞
Tarantula	40.3	1.97	0.1
Ochiai	41.0	1.98	0.1
DStar	41.5	1.99	0.1
Random	39.2	1.92	0.1



6 Sonuç

YSA'ların güvenlik öncelikli sistemlerde kullanılmaya başlaması, kabul edilemez riskleri de beraberinde getirmektedir. Bu riskleri minimum seviyelere indirebilmek için, YSA'ların derinlemesine test edilmesi, kritik önem taşımaktadır. Bu makalede, bu çabaya katkıda bulumak adına, daha önceden eğitilmiş, ileri beslemeli yapay sinir ağlarını Java kaynak koduna çevirdikten sonra JaCoCo kod kapsamı aracını çalıştırıp, her nöronun isabet izgesine ulaşan, şüphelilik ölçütleri kullanarak şüpheli nöronları tespit eden ve şüpheli nöronların aktivasyon değerlerini arttırarak yeni girdiler sentezleyen tekniğimizden bahsettik. En çok kullanılan veri kümelerinden MNIST'i kullanarak yaptığımız deneysel değerlendirmelerimiz, yetersiz performansa neden olan nöronların, tekniğimiz sayesinde tespit edilebildiğini göstermiştir. Tekniğimiz ayrıca, orijinaline çok benzeyen, yüksek hasımlık taşıyan ve tespit edilen şüpheli nöronları harekete geçiren yeni girdiler yaratmamıza olanak sağlamıştır. Gelecekte, tekniğimizi farklı YSA'lar ve veri kümeleri üzerinde denemeyi ve şüphelilik-güdümlü sentezleme algoritmamızı geliştirmeyi planlıyoruz. Ayrıca, tespit edilen şüpheli nöronları onarabilen teknikleri keşfetmeye çalışacağız.

Kaynakça

1. Burton, S., Gauerhof, L., Heinzemann, C.: Making the case for safety of machine learning in highly automated driving. In: Computer Safety, Reliability, and Security. pp. 5–16 (2017)
2. Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., et al.: AI2: Safety and robustness certification of neural networks with abstract interpretation. In: IEEE Symposium on Security and Privacy (S&P). pp. 1–16 (2018)
3. Gopinath, D., Wang, K., Zhang, M., Pasareanu, C.S., Khurshid, S.: Symbolic execution for deep neural networks. In: arXiv preprint arXiv:1807.10439 (2018)
4. Huang, X., Kwiatkowska, M., Wang, S., Wu, M.: Safety verification of deep neural networks. In: International Conference on Computer Aided Verification (CAV). pp. 3–29 (2017)
5. Jones, J.A., Harrold, M.J.: Empirical evaluation of the tarantula automatic fault-localization technique. In: IEEE/ACM International Conference on Automated Software Engineering (ASE). pp. 273–282 (2005)
6. Jorgensen, P.C.: Software testing: a craftsman's approach. Auerbach Publications (2013)

7. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient SMT solver for verifying deep neural networks. In: International Conference on Computer Aided Verification (CAV). pp. 97–117 (2017)
8. Kim, J., Feldt, R., Yoo, S.: Guiding deep learning system testing using surprise adequacy. In: arXiv preprint arXiv:1808.08444 (2018)
9. Landsberg, D., Chockler, H., Kroening, D., Lewis, M.: Evaluation of measures for statistical fault localisation and an optimising scheme. In: International Conference on Fundamental Approaches to Software Engineering (FASE). pp. 115–129 (2015)
10. LeCun, Y.: The MNIST database of handwritten digits. http://yann. lecun. com/exdb/mnist (1998)
11. Ma, L., Juefei-Xu, F., Zhang, F., Sun, J., et al.: DeepGauge: Multi-granularity testing criteria for deep learning systems. In: IEEE/ACM International Conference on Automated Software Engineering (ASE) (2018)
12. Ma, L., Zhang, F., Sun, J., Xue, M., et al.: DeepMutation: Mutation testing of deep learning systems. In: IEEE International Symposium on Software Reliability Engineering (ISSRE) (2018)
13. Ma, L., Zhang, F., Xue, M., Li, B., et al.: Combinatorial testing for deep learning systems. In: arXiv preprint arXiv:1806.07723 (2018)
14. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: International Conference on Machine Learning (ICML). vol. 30, p. 3 (2013)
15. Ochiai, A.: Zoogeographic studies on the soleoid fishes found in japan and its neighbouring regions. Bulletin of Japanese Society of Scientific Fisheries **22**, 526–530 (1957)
16. Pearson, S., Campos, J., Just, R., Fraser, G., et al.: Evaluating and improving fault localization. In: International Conference on Software Engineering (ICSE). pp. 609–620 (2017)
17. Pei, K., Cao, Y., Yang, J., Jana, S.: DeepXplore: Automated whitebox testing of deep learning systems. In: Symposium on Operating Systems Principles (SOSP). pp. 1–18 (2017)
18. Seshia, S.A., Desai, A., Dreossi, T., Fremont, D.J., Ghosh, S., Kim, E., Shivakumar, S., Vazquez-Chanlatte, M., Yue, X.: Formal specification for deep neural networks. In: Technical Report, University of California at Berkeley (2018)
19. Sun, Y., Wu, M., Ruan, W., Huang, X., et al.: Concolic testing for deep neural networks. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE). pp. 109–119 (2018)
20. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., et al.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
21. Tian, Y., Pei, K., Jana, S., Ray, B.: Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In: International Conference on Software Engineering (ICSE). pp. 303–314 (2018)
22. Wicker, M., Huang, X., Kwiatkowska, M.: Feature-guided black-box safety testing of deep neural networks. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). pp. 408–426 (2018)
23. Wong, W.E., Debroy, V., Gao, R., Li, Y.: The DStar method for effective software fault localization. IEEE Transactions on Reliability **63**(1), 290–308 (2014)
24. Wong, W.E., Gao, R., Li, Y., Abreu, R., Wotawa, F.: A survey on software fault localization. IEEE Transactions on Software Engineering **42**(8), 707–740 (2016)
25. Wu, M., Wicker, M., Ruan, W., Huang, X., Kwiatkowska, M.: A game-based approximate verification of deep neural networks with provable guarantees. In: arXiv preprint arXiv:1807.03571 (2018)

Aşırı Öğrenme Makinesi Algoritmasını Kullanarak Yazılım Zafiyet Kestirimi

Ali Seydi Keçeli[1], Aydın Kaya[1], Çağatay Çatal[2], Bedir Tekinerdoğan[2]

[1] Hacettepe Üniversitesi, Bilgisayar Mühendisliği, Türkiye
{aliseydi, aydinkaya}@cs.hacettepe.edu.tr
[2] Wageningen Üniversitesi, Bilgi Teknolojileri Grubu, Hollanda
{cagatay.catal, bedir.tekinerdogan}@wur.nl

Özet. Yazılım zafiyet kestirimi, yazılımın operasyonel ortama konuşlandırılmadan önce kaynak kod içerisindeki zafiyetlerinin tespit edilmesini amaçlamaktadır. Zafiyetlerin doğru şekilde kestirilmesi daha fazla test kaynağının zafiyet-eğilimli modüllere ayrılmasına yardımcı olmaktadır. Makine öğrenmesi bakışı açısıyla, bu problem yazılım modüllerini, zafiyet-eğilimli ve zafiyet-eğilimli-olmayan seklinde iki kategoriye ayıran ikili bir sınıflandırma işlemidir. Yazılım zafiyet kestirimi için makine öğrenmesi temelli modeller şimdiye kadar geliştirilmiştir ancak su ana kadar teknikte gelinen en son noktada modellerin performansı kabul edilebilir seviyede değildir. Bu çalışmada, yazılım zafiyet kestirim modellerinin performanslarını iyileştirmek üzere bu problemde henüz çalışılmamış olan Aşırı Öğrenme Makinesi algoritmalarından yararlanarak performansı iyileştirmeyi amaçlıyoruz. AÖM algoritmalarını 3 açık veri kümesinde uygulamadan önce, iki sınıfa ait olan veri noktalarını dengelemek üzere veri dengeleme algoritmalarını kullanıyoruz. Bu kapsamda yaptığımız deneylerle ilgili ilk deneysel sonuçlarımızı tartışarak öğrenilen dersleri sunuyoruz. AÖM algoritmalarının yazılım zafiyet kestirimi probleminde yüksek potansiyele sahip olduğu gözlemlenmektedir.

Anahtar Kelimeler: Yazılım Zafiyet Kestirimi, Makine Öğrenmesi, Aşırı Öğrenme Makinesi

Software Vulnerability Prediction using Extreme Learning Machines Algorithm

Ali Seydi Keçeli[1], Aydın Kaya[1], Çağatay Çatal[2], Bedir Tekinerdoğan[2]

[1] Hacettepe University, Computer Engineering, Turkey
{aliseydi, aydinkaya}@cs.hacettepe.edu.tr
[2] Wageningen University, Information Technology Group, Holland
{cagatay.catal, bedir.tekinerdogan}@wur.nl

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Abstract. Software vulnerability prediction aims to detect vulnerabilities in the source code before the software is deployed into the operational environment. The accurate prediction of vulnerabilities helps to allocate more testing resources to the vulnerability-prone modules. From the machine learning perspective, this problem is a binary classification task which classifies software modules into vulnerability-prone and non-vulnerability-prone categories. Several machine learning models have been built for addressing the software vulnerability prediction problem, but the performance of the state-of-the-art models is not yet at an acceptable level. In this study, we aim to improve the performance of software vulnerability prediction models by using Extreme Learning Machines (ELM) algorithms which have not been investigated for this problem. Before we apply ELM algorithms for selected three public datasets, we use data balancing algorithms to balance the data points which belong to two classes. We discuss our initial experimental results and provide the lessons learned. In particular, we observed that ELM algorithms have a high potential to be used for addressing the software vulnerability prediction problem.

Keywords: Software Vulnurability Prediction, Machine Learning, Extreme Learning Machines

1 Giriş

Büyük ölçekteki organizasyonları, şirketleri veya KOBİ'leri etkileyebilecek çok sayıda ve farklı seviyelerde zafiyetten söz etmek mümkündür. Örneğin, bu zafiyetler ağ seviyesinde, personel seviyesinde, donanım seviyesinde olabileceği gibi kullanılmakta olan uygulama veya sistem yazılımlarından kaynaklı olarak yazılım seviyesinde de olabilmektedir. Zafiyetlerin doğru zamanda ve etkin şekilde tespit edilmediği durumda, gerek işlenmekte ve saklanmakta olan veri açısından gerekse de sistemin işleyişi açısından, gizlilik, mahremiyet, bütünlük gibi birçok kalite faktörü kapsamında sorunlar yaşanmaktadır.

Günümüzde geliştirilmekte olan birçok yazılım sistemi, son yıllarda kaynak kod satır sayısı açısından birkaç milyon satır düzeyine ulaşmış durumdadır ve önceden beri organizasyon içerisinde var olan sistemlerle (legacy systems) olan entegrasyonunu da dikkate aldığımız zaman, tipik olarak 50 milyon veya 100 milyon kod satırı düzeyinde yazılımlar karşımıza çıkabilmektedir. Dünyadaki genel trendlere bakıldığında bu tur çok geniş ölçekli yazılım sistemlerinin, onumuzdaki 20-30 yıl suresince ultra geniş ölçekli sistemler haline gelerek, milyon kod satır sayısından milyar kod satırına doğru değişim göstereceği, Carnegie Mellon Üniversitesi, Yazılım Mühendisliği Enstitüsü (SEI) tarafından hazırlanan raporlarda ifade edilmektedir.

Kaynak kod satır sayısının gittikçe artmakta olduğu bu bağlamda, zafiyetlerin ortaya çıkmadan önce kestirilebilmesi gittikçe önem arz eden bir husus haline gelmiştir. Tek bir yazılım dahi, artık tek başına çalışmaktan çok öte, bulunduğu platformdaki diğer altyapılarla koordine ve eş güdüm halinde çalışmak durumundadır. Bu altyapılar; farklı yazılım çerçeveleri, ara katmanlar, veritabanı yönetim sistemleri, yazılım kütüphaneleri şeklinde olabileceği gibi gelişmekte olan farklı teknolojilerden kaynaklı blok zinciri altyapısı, bağlı veri platformu, büyük veri işleme sistemi gibi özelleşmiş

düzeyde de olabilmektedir. Tek bir yazılım sisteminin çalışmasını incelerken dahi, etkileşimde olduğu bu tür diğer bileşenleri dikkate aldığımız zaman, her bir bileşenden bir güvenlik tehlikesi veya zafiyeti oluşması oldukça muhtemeldir. Güvenli yazılım geliştirme yaşam çevrimleri, güvenli yazılım tasarımı, statik ve dinamik yazılım analiz araçları, uygulama seviyesindeki güvenlik duvarları bazı sorunları ortadan kaldırsa da bu tür teknikler ve teknolojilerle tamamen tüm güvenlik sorunlarının çözülebildiğini söyleyebilmek söz konusu değildir, aksi halde hemen hemen her gün karşımıza çıkan siber saldırılar, güvenlik ihlalleri ve yetkisiz kişilerin yazılım sistemleri üzerinde denetim kazanarak çıkar veya kazanç elde edilebilmesi mümkün olmazdı.

Sızma testleri (penetration testing) ve statik kod analiz araçları açısından zafiyet tespitini dikkate aldığımız zaman, yanlış pozitif oranının oldukça yüksek olduğu bilinmektedir. Kod gözden geçirme (code review) alternatif bir yöntem seklinde değerlendirilse de, 50-100 milyon kod satırı düzeyindeki yazılımlar için tüm bileşenler bağlamında bu işlemin gerçekleştirilmesi, kaynak ve bütçe açısından çoğu durumda uygulanabilir olmamaktadır.

Bu kısıtlar nedeniyle, son dönemde bazı araştırmacılar makine öğrenmesi algoritmalarından yararlanarak, zafiyet-eğilimli modülleri kestirerek, doğrulama ve geçerleme kaynaklarını bu modüllere daha fazla ayırmakta ve bu sayede, zafiyetleri azaltmayı hedeflemektedir. Bu araştırma alanı, yazılım zafiyet kestirimi olarak adlandırılmakta olup, yazılım kusur kestirimi (software fault prediction) problemiyle kullanılabilecek metrikler ve yöntemler açısından benzerlik göstermektedir ancak zafiyetlerin, kusurlara kıyasla çok daha küçük bir alt küme olması ancak zafiyetlerin ortaya çıkması durumunda, etkilerinin çok daha kritik seviyede olması ve bu zafiyetleri tespit açısından, kaynak kodun ve saldırı örüntülerinin daha derin seviyede bilinmesini gerektirmesi açısından çok daha farklı bir problem olduğunu ifade etmek mümkündür. Araştırmacı bakış açısıyla diğer bir zorluk ise şirketlerin genel olarak zafiyetlere ilişkin olarak verilerini paylaşmak istememesi nedeniyle, bu alanda kamuya açık veri kümesi üretilmesi ve paylaşılması çok fazla görünmemektedir. Bu çalışmada, Stuckman ve arkadaşlarının [3] oluşturulmuş olduğu 3 adet kamuya açık veri kümesinden yararlanılmıştır.

Bu çalışmanın genel amacı, mevcut durumda istenen düzeyde olmayan yazılım zafiyet kestirimi modellerinin performansını daha iyi hale getirmek ve bu amaçla, veri dengeleme (data balancing) ve Aşırı Öğrenme Makineleri (Extreme Learning Machines-AÖM) algoritmalarından yararlanmaktır. AÖM algoritmalarının daha hızlı öğrenme hızı, insan müdahalesi açısından daha az gereksinim ihtiyacı ve diğer problemlerde raporlanmış olan daha iyi genelleştirme yeteneği nedeniyle, bu problemde de ilk kez bu çalışma kapsamında incelenmesi hedeflenmiştir. Bu alandaki veri kümelerinin dengesiz (unbalanced) olması nedeniyle (örneğin, %90 oranında zafiyet içermeyen veri noktasının oluşturduğu bir sınıf ve %10 oranında zafiyet içeren veri noktalarının olduğu başka bir sınıf), veri dengeleme algoritmalarının da model performansını iyileştirme açısından değerlendirilmesi planlanmıştır.

Bu çalışmanın literatüre temel katkısı; veri dengeleme algoritmalarının ve Aşırı Öğrenme Makinesi algoritmalarının, yazılım zafiyet kestirimi problemi bağlamında ilk kez uygulanarak gelecek açısından ümit verici sonuçlarının elde edilmiş olmasıdır.

Takip eden bölüm 2'de ilişkili çalışmalar sunulmuştur. Bölüm 3'de önerilmekte olan yöntem açıklanmaktadır. Bölüm 4'de deneysel sonuçlar verilerek Bölüm 5'de sonuçlar ve gelecek çalışmalar ortaya konulmuştur.

2 İlişkili Çalışmalar

Yazılım zafiyet kestirimi, yazılım güvenliği ile doğrudan ilgili ve yazılım kalitesini etkileyen kalite güvence aktivitelerinden birisidir. Yazılım zafiyet çalışmaları büyük oranda makine öğrenmesi yaklaşımlarından faydalanmaktadır. Geçmiş veriye bağlı olarak oluşturulmuş bu kestirimsel modeller, yazılım test aşamasından önce çalıştırılarak, zafiyet eğilimli modüller ile ilgili sınıflandırmalar gerçekleştirilmektedir. Bu sonuçlar ışığında, zafiyet eğilimli olarak etiketlenen modüller daha detaylı incelemelere tabi tutulmaktadırlar.

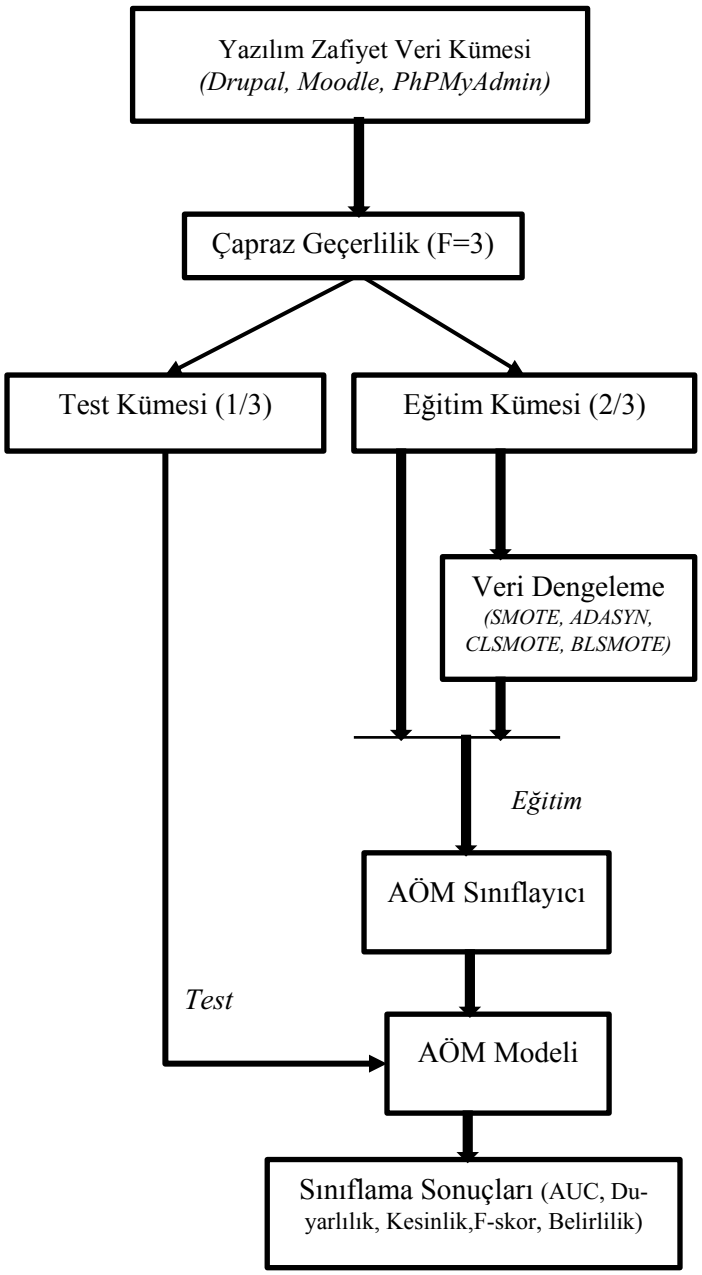
Yazılım zafiyet kestiriminde (YZK) genel olarak yazılım metrikleri ve yazılım kaynak kodlarından üretilmiş metin öznitelikleri kullanılmaktadır [1]. Jimenez ve arkadaşları tarafından yapılan çalışmada; yazılım metrikleri, metin madenciliği ve istatistiki analiz uyarı yaklaşımları birleştirilmiştir [2]. Yazılım metrikleri ve metin özniteliklerinden faydalanan bir başka çalışma Stuckman ve arkadaşları [3] tarafından önerilmiştir. Bu çalışmada ayrıca birkaç veri dengeleme yaklaşımlarının yazılım zafiyet kestirimi üzerine etkileri de incelenmiştir. Largerstorm ve arkadaşları [4] farklı yazılım metriklerinin zafiyet tespiti üzerindeki etkilerini incelemiştir. Bileşen ve mimari seviyedeki metriklerin, zafiyet tespitinde etkili oldukları görülmüştür.

Shin ve arkadaşları [5] kaynak kod değişim (*code chunk*), karmaşıklık ve geliştirici aktivitelerinin yazılım zafiyet kestirimi üzerindeki etkilerini incelemiştir. Walden ve arkadaşları [6] tarafından yapılan çalışmada metin madenciliği ile üretilen özniteliklerin yazılım metriklerine göre daha başarılı sonuçlar ürettiği gözlemlenmiştir. Sultana [7], istatistiki yöntemler ile makine öğrenmesi yöntemlerini birlikte kullanan modeller önermiştir. Shin ve Williams [8] yazılım hata tespiti modellerinin zafiyet tespitinde de kullanılıp kullanılamayacağı üzerine testler yapmışlardır. Elde edilen sonuçlar ışığında zafiyet ve hata tespiti modellerinin yakın sonuçlar ürettiği gözlemlenmiştir. Morrison ve arkadaşları [9] Windows işletim sistemi üzerinde çalıştırılmak üzere 2 YZK modeli önermiştir. Pang ve arkadaşları [10] derin öğrenme ağı kullanarak zafiyet kestirimi gerçekleştirmişlerdir. Derin öğrenme ağının eğitiminde kaynak kodlardan elde edilen öznitelikler kullanılmıştır. Yaptığımız literatür taramasına göre, Aşırı Öğrenme Makinesi (AÖM) algoritmaları bu problem özelinde henüz uygulanmamış olup ilk kez bu yayınlamakta olduğumuz çalışma kapsamında ele alınmıştır.

3 Yöntem

Yönteme ilişkin genel akış şeması Şekil-1 de gösterilmiştir. İlk adım olarak veri seti test ve eğitim kümelerine ayrılmıştır. Sonraki adımda eğitim kümesi üzerinde dengeleme yapılarak ve yapılmadan AÖM modelleri oluşturulmuştur. Oluşturulan bu modeller üzerinde test kümesi kullanılarak sınıflandırma yapılmış ve sınıflandırma sonuçlarına ait AUC (Area Under ROC Curve-ROC eğrisi altında kalan alan), duyarlık,

keskinlik, F-skor ve belirlilik ölçümleri yapılmıştır. Deneyler sırasında üçlü çapraz geçerlik testi uygulanmıştır. Çapraz geçerlik testinde; veri kümesi rastgele 3 parçaya ayrılmakta ve parçalardan 2'si eğitim, kalan tek parça ise test için kullanılmaktadır.



Şekil 1. Uygulanan deney tasarımının genel akışı.

3.1 Veri Dengeleme

Yazılım zafiyet tespiti için kullanılan veri kümeleri son derece dengesiz olması sebebiyle model eğitiminden önce çeşitli veri dengeleme yöntemleri ile daha homojen bir eğitim kümesi oluşturulmuştur. Veri dengeleme için SMOTE [11], CLSMOTE [12], BLSMOTE [13] ve ADASYN [14] yöntemleri kullanılmıştır. SMOTE ve türevleri var olan örnekleri çoğaltmak yerine yeni örnekler üretmektedir. CLSMOTE ve BLSMOTE, SMOTE yaklaşımının geliştirilmiş sürümleridir. CLSMOTE yaklaşımında veri önce kümelenmekte sonrasında ise standart SMOTE uygulanmaktadır. BLSMOTE yönteminde ise sınır (borderline) örnekler ilk olarak tespit edilmektedir. Yeni üretilecek örnekler, bu sınır örnekler üzerinden tespit edilmektedir. Kullanılan son dengeleme yöntemi ise ADASYN'dir. Bu yaklaşımda, azınlık sınıfları için farklı ağırlıklarda dağılımlar kullanılmaktadır.

3.2 Aşırı Öğrenme Makinesi

Aşırı öğrenme makineleri (AÖM) sadece tek gizli katmandan oluşan ileri beslemeli bir sinir ağı (neural network) modelidir [15]. Hem sınıflama hem de regresyon problemlerinde kullanılabilmektedir. Örnek bir AÖM modeli Şekil-2'de gösterilmiştir. Şekil-2'de w girdi bağlantılarının ağırlıklarını β çıktı ağırlıklarını ve g aktivasyon fonksiyonunu belirtmektedir. Gizli katmandaki aktivasyon işlemleri Eşitlik-1 ile ifade edilebilir.

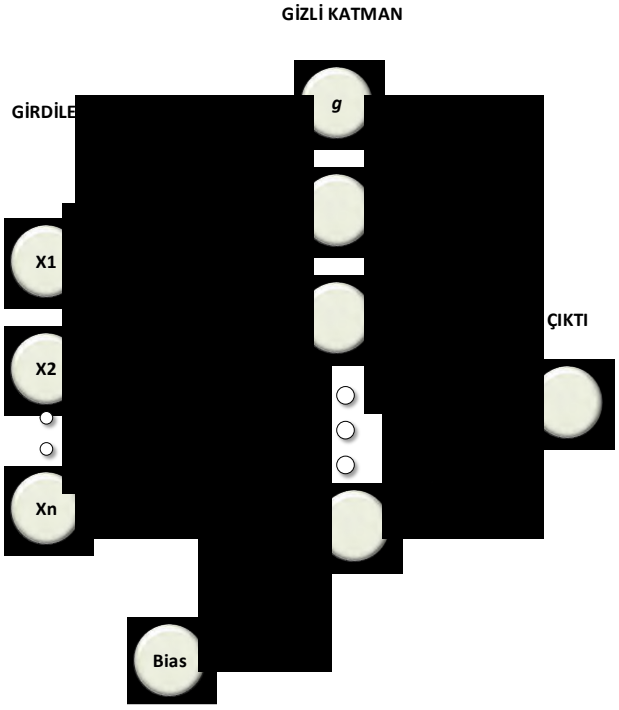
$$\sum_{j=1}^{M} \beta_i \, g(w_j x_j + b_j) \tag{1}$$

Geleneksel yapay sinir ağı yaklaşımlarından farklı olarak giriş katmanına uygulanan ağırlıklar rastgele bir şekilde verilmektedir. AÖM'ye ait rastgele olarak atanan bu değerler daha sonra eğitim aşamasında güncellenmemektedir. Buna karşın gizli katman ve çıktı katmanı arasındaki bağlantılara atanan ağırlık değerleri doğrusal bir model yardımıyla analitik olarak hesaplanır. Geleneksel sinir ağlarında katmanlar arasındaki ağırlıklar iteratif olarak yani hatayı en az yapan değerlere ulaşılana kadar farklı değerlerin sınanmasıyla hesaplanırken, AÖM'lerde bu ağırlıkların doğrusal olarak hesaplanması modelin eğitiminde ciddi bir hızlanmaya sebep olmaktadır [15]. AÖM yapısında çıkış katmanı Eşitlik-2'deki hale dönüştürülüp Eşitlik-3 şeklinde ifade edilebilir. Eşitlik-3'teki ifade artık doğrusal olarak çözülebilir.

$$H(w_{i,j}, x_i, b_i) = \begin{bmatrix} g(w_{1,1}, x_1, b_1) & \cdots & g(w_{1,m}, x_m, b_m) \\ \vdots & \ddots & \vdots \\ g(w_{n,1}, x_1, b_1) & \cdots & g(w_{n,m}, x_m, b_m) \end{bmatrix} \tag{2}$$

$$y = H\,\beta \tag{3}$$

Ayrıca yapılan çalışmalarda AÖM yaklaşımının pek çok uygulama alanında yaygın bir diğer makine öğrenmesi yöntemi olan SVM yaklaşımından daha başarılı olduğu görülmüştür [16, 17].



Şekil 2. Aşırı Öğrenme Makinesi

Geleneksel sinir ağlarından daha az parametreye sahip olması ve eğitim parametrelerinin belirlenmemesi gibi avantajlara sahiptir. Ancak ara katmanın nöron sayısı ve kullanılacak olan aktivasyon fonksiyonu seçimi, kullanıcı tarafından belirlenmektedir. Aktivasyon fonksiyonları bir nörona gelen girdiye uygulanan ve bu bağlamda oluşturulacak çıktıyı hesaplayan fonksiyonlardır. Bu modellerin doğrusal olmama özelliği, aktivasyon fonksiyonlarının da doğrusal olmamasından kaynaklanmaktadır. AÖM tarafından kullanılan aktivasyon fonksiyonları Tablo-1'de belirtilmiştir.

Tablo 1. AÖM'de kullanılan bazı aktivasyon fonksiyonları [18]

Aktivasyon Fonksiyonu	*Eşitlik*
Sigmoid	$a = \frac{1}{1 + \exp(-n)}$
Radyal Bazlı	$a = \exp(-n^2)$
Normalleştirilmiş Radyal Bazlı	$a = \frac{\exp(-n^2)}{\sum \exp(-n^2)}$
Elliot Sigmoid	$a = \frac{0.5 * \mathrm{n}}{1 + abs(n)} + 0.5$
Hard Limit	$a = \begin{cases} 1, eğer\ n \geq 0 \\ 0, diğer \end{cases}$
Sinüs	$\mathrm{a} = \sin(e)$
Kosinüs	$\mathrm{a} = \cos(e)$
Dogrusal	$a = n$
SoftMax	$a = \frac{\exp(n)}{\sum \exp(n)}$
Ters	$a = \frac{1}{n}$
Pozitif Doğrusal	$a = \begin{cases} n, eğer\ n \geq 0 \\ 0, n < 0 \end{cases}$

4 Deneysel Sonuçlar

Bu bölümde, Sekil 1'de detayları verilmekte olan deney tasarımına ilişkin tüm deneysel sonuçlar sunulmaktadır.

Yazılım metriklerinin nitelik olarak kullanıldığı makine öğrenmesi modellerine ilişkin sonuçlar, Tablo 2-3-4'de sunulmuştur. Bu kapsamda kullanılan yazılım metrikleri; HTML içermeyen kod satır sayısı, PHP dosyası içerisindeki kod satır sayısı, metod sayısı, çevrimsel karmaşıklık (cyclomatic complexity), maksimum iç içe geçme karmaşıklığı (maximum nesting complexity), Halstead hacim, fan-in, fan-out, çağrılan iç metotların sayısı, toplam dış çağrıların sayısı, dış metotların sayısı, metotlara olan dış çağrıların sayısı olarak 12 tanedir. Kamuya açık veri kümelerinde bu metriklerin detayları açıklanmaktadır.

Yazılım kaynak kodunun metin olarak değerlendirilerek oluşturulan sözcük frekans analizi temelli veri kümelerine ilişkin analiz sonuçları ise Tablo 5-6-7'de verilmektedir. Kaynak kod içerisindeki her bir belirteç (token) bir nitelik olarak veri kümelerinde temsil edilmektedir. Tablo 8-9-10'da ise hem yazılım metriklerinin hem de metinsel niteliklerin birleştirildiği durumdaki veri kümelerinde (bütünleşik) yapılan analizlerin sonuçları sunulmuştur.

AUC değerlendirme parametresi referans alınarak yukarıda belirtilen tablolar incelendiğinde; metin seviyesindeki niteliklerle kurulan modellerin, yazılım metrik seviyesindeki niteliklerle kurulan modellerden daha iyi performans verdiği, bütünleşik modellerin performansıyla metin seviyesindeki modellerin performansı kıyaslandığı durumda bütünleşik nitelikler seviyesinde performansın çoğu durumda biraz daha iyi olduğu ancak aradaki farkın çok da fazla olmadığı gözlemlenmiştir. Bu nedenle, sadece metin seviyesindeki niteliklerle model kurulmasının daha uygun olacağı önerilmektedir.

Metin seviyesindeki nitelikler ve bütünleşik seviyedeki niteliklerle elde edilen performanslar incelendiğinde, çoğu durumda ADASYN isimli veri dengeleme yöntemiyle daha başarılı kestirim modellerin kurulabildiği gözlemlenmiştir. Bu nedenle, veri dengeleme algoritmalarını da kestirim modellerine eklemek isteyen araştırmacılar için ADASYN algoritması önerilmektedir.

Tablo 2. Drupal metrik veri kümesi için sınıflama sonuçları

Dengeleme	AUC	Duyarlılık	Kesinlik	F-skor	Belirlilik
Dengelenmemiş	0.6206	0.4242	0.6077	0.4988	0.8777
ADASYN	0.6210	0.6594	0.5386	0.5923	0.7501
SMOTE	0.6166	0.6406	0.5455	0.5889	0.7637
BL SMOTE	0.6232	0.6574	0.5336	0.5886	0.7451
CL SMOTE	0.6159	0.6300	0.5311	0.5760	0.7536

Tablo 3. Moodle metrik veri kümesi için sınıflama sonuçları

Dengeleme	AUC	Duyarlılık	Kesinlik	F-skor	Belirlilik
Dengelenmemiş	0.6072	0.0000	NaN	NaN	0.9997
ADASYN	0.6171	0.5200	0.0225	0.0431	0.8145
SMOTE	0.6204	0.5108	0.0225	0.0432	0.8182
BL SMOTE	0.6173	0.5292	0.0233	0.0446	0.8177
CL SMOTE	0.6242	0.5317	0.0237	0.0454	0.8202

Tablo 4. PhPMyadmin metrik veri kümesi için sınıflama sonuçları

Dengeleme	AUC	Duyarlılık	Kesinlik	F-skor	Belirlilik
Dengelenmemiş	0.6641	0.1526	0.6019	0.2402	0.9902
ADASYN	0.6989	0.3289	0.1437	0.1998	0.8203
SMOTE	0.6960	0.2970	0.1396	0.1896	0.8327
BL SMOTE	0.7015	0.3081	0.1405	0.1928	0.8264
CL SMOTE	0.6987	0.3163	0.1477	0.2009	0.8331

Tablo 5. Drupal metin veri kümesi için sınıflama sonuçları

Dengeleme	AUC	Duyarlılık	Kesinlik	F-skor	Belirlilik
Dengelenmemiş	0.7071	0.4732	0.5552	0.5091	0.8311
ADASYN	0.7152	0.4858	0.5320	0.5070	0.8100
SMOTE	0.7115	0.4871	0.5372	0.5097	0.8136
BL SMOTE	0.7132	0.4913	0.5256	0.5068	0.8034
CL SMOTE	0.7125	0.4784	0.5456	0.5089	0.8234

Tablo 6. Moodle metin veri kümesi için sınıflama sonuçları

Dengeleme	AUC	Duyarlılık	Kesinlik	F-skor	Belirlilik
Dengelenmemiş	0.7218	0.0708	0.2119	NaN	0.9978
ADASYN	0.7325	0.2042	0.0314	0.0543	0.9481
SMOTE	0.7287	0.2142	0.0322	0.0560	0.9472
BL SMOTE	0.7336	0.2167	0.0325	0.0565	0.9470
CL SMOTE	0.7247	0.2450	0.0318	0.0562	0.9384

Tablo 7. PhPMyadmin metin veri kümesi için sınıflama sonuçları

Dengeleme	AUC	Duyarlılık	Kesinlik	F-skor	Belirlilik
Dengelenmemiş	0.7510	0.1748	0.1922	NaN	0.9327
ADASYN	0.7654	0.2593	0.1057	0.1499	0.7981
SMOTE	0.7622	0.2733	0.1072	0.1536	0.7927
BL SMOTE	0.7574	0.2852	0.1144	0.1630	0.7977
CL SMOTE	0.7509	0.2741	0.1156	0.1623	0.8077

Tablo 8. Drupal bütünleşik (metin + metrik) veri kümesi için sınıflama sonuçları

Dengeleme	AUC	Duyarlılık	Kesinlik	F-skor	Belirlilik
Dengelenmemiş	0.7002	0.5026	0.5527	0.5250	0.8196
ADASYN	0.7181	0.4861	0.5160	0.4991	0.7971
SMOTE	0.7107	0.5003	0.5286	0.5127	0.8013
BL SMOTE	0.7084	0.5126	0.5208	0.5151	0.7894
CL SMOTE	0.7091	0.4961	0.5256	0.5093	0.8019

Tablo 9. Moodle bütünleşik (metin + metrik) veri kümesi için sınıflama sonuçları

Dengeleme	AUC	Duyarlılık	Kesinlik	F-skor	Belirlilik
Dengelenmemiş	0.7276	0.0675	0.2370	NaN	0.9981
ADASYN	0.7495	0.1908	0.0241	0.0427	0.9365
SMOTE	0.7441	0.1942	0.0243	0.0432	0.9356
BL SMOTE	0.7414	0.1983	0.0244	0.0434	0.9344
CL SMOTE	0.7346	0.2233	0.0245	0.0441	0.9269

Tablo 10. PhPMyadmin bütünleşik (metin + metrik) veri kümesi sınıflama sonuçları

Dengeleme	AUC	Duyarlılık	Kesinlik	F-skor	Belirlilik
Dengelenmemiş	0.7282	0.1519	0.2377	0.1835	0.9561
ADASYN	0.7446	0.2630	0.1271	0.1709	0.8359
SMOTE	0.7385	0.2578	0.1341	0.1756	0.8472
BL SMOTE	0.7367	0.2837	0.1342	0.1817	0.8329
CL SMOTE	0.7356	0.2570	0.1334	0.1752	0.8478

5 Sonuç ve Gelecek Çalışmalar

Bu çalışmada, yazılım zafiyet kestirimi probleminde veri dengeleme algoritmaları Aşırı Öğrenme Makinesi algoritmalarıyla birlikte ilk kez uygulanarak, metrik, metin ve bütünleşik (metin+metrik) nitelik modellerinden yararlanılarak 3 farklı kamuya açık veri kümesinde çok sayıda deney gerçekleştirilmiştir. Elden edilen deneysel sonuçlara göre, Aşırı Öğrenme Makinesi algoritmalarının bu problem özelinde potansiyele sahip olduğu sonucuna varılmış olup daha iyi performans elde edebilmek üzere ek araştırmaların yapılması planlanmıştır. Bu kapsamda, gelecek çalışmalarda daha farklı veri dengeleme algoritmalarının incelenmesi ve ek olarak daha farklı Aşırı Öğrenme Makinesi algoritmalarının deneylere entegre edilmesi söz konusu olabilecektir. Ayrıca, daha fazla veri kümesi elde edilmesi mümkün olduğu durumda, deneyler daha fazla sayıda veri kümesinde gerçekleştirilebilecektir.

Kaynakça

1. Hovsepyan, A., Scandariato, R., Joosen, W., Walden, J.: Software vulnerability prediction using text analysis techniques. In: Proceedings of the 4th international workshop on Security measurements and metrics, pp. 7-10. ACM, (2012)
2. Jimenez, M., Papadakis, M., Le Traon, Y.: Vulnerability prediction models: A case study on the linux kernel. In: 2016 IEEE 16th International Working Conference on Source Code Analysis and Manipulation (SCAM), pp. 1-10. IEEE, (2016)

3. Stuckman, J., Walden, J., Scandariato, R.: The Effect of Dimensionality Reduction on Software Vulnerability Prediction Models. Ieee T Reliab 66, 17-37 (2017)
4. Lagerström, R., Baldwin, C., MacCormack, A., Sturtevant, D., Doolan, L.: Exploring the relationship between architecture coupling and software vulnerabilities. In: International Symposium on Engineering Secure Software and Systems, pp. 53-69. Springer, (2017)
5. Shin, Y., Meneely, A., Williams, L., Osborne, J.A.: Evaluating Complexity, Code Churn, and Developer Activity Metrics as Indicators of Software Vulnerabilities. Ieee T Software Eng 37, 772-787 (2011)
6. Walden, J., Stuckman, J., Scandariato, R.: Predicting Vulnerable Components: Software Metrics vs Text Mining. Proc Int Symp Softw 23-33 (2014)
7. Sultana, K.Z.: Towards a Software Vulnerability Prediction Model using Traceable Code Patterns and Software Metrics. Ieee Int Conf Autom 1022-1025 (2017)
8. Shin, Y., Williams, L.: Can traditional fault prediction models be used for vulnerability prediction? Empir Softw Eng 18, 25-59 (2013)
9. Morrison, P., Herzig, K., Murphy, B., Williams, L.: Challenges with applying vulnerability prediction models. In: Proceedings of the 2015 Symposium and Bootcamp on the Science of Security, pp. 4. ACM, (2015)
10.Pang, Y., Xue, X., Wang, H.: Predicting vulnerable software components through deep neural network. In: Proceedings of the 2017 International Conference on Deep Learning Technologies, pp. 6-10. ACM, (2017)
11.Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: Synthetic minority over-sampling technique. J Artif Intell Res 16, 321-357 (2002)
12.Cieslak, D.A., Chawla, N.V., Striegel, A.: Combating imbalance in network intrusion datasets. 2006 Ieee International Conference on Granular Computing 732-737 (2006)
13.Han, H., Wang, W.Y., Mao, B.H.: Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. Advances in Intelligent Computing, Pt 1, Proceedings 3644, 878-887 (2005)
14.He, H.B., Bai, Y., Garcia, E.A., Li, S.T.: ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning. 2008 Ieee International Joint Conference on Neural Networks, Vols 1-8 1322-1328 (2008)
15.Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: Theory and applications. Neurocomputing 70, 489-501 (2006)
16.Huang, G.-B.: What are extreme learning machines? Filling the gap between Frank Rosenblatt's dream and John von Neumann's puzzle. Cogn Comput 7, 263-278 (2015)
17.Huang, G.B.: An Insight into Extreme Learning Machines: Random Neurons, Random Features and Kernels. Cogn Comput 6, 376-390 (2014)
18.Özçalici, M.: Aşiri Öğrenme Makineleri İle Hisse Senedi Fiyat Tahmini. Hacettepe Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi 35, 67-88 (2017)

Yapay Zeka Tabanlı Yüz Tanıma Sisteminin Geliştirilmesi Ve Optimizasyonu

Sedat Yıldız[1], Erdem Özgür[1], Nihal Bilal[1],

[1]Teracity Yazılım Teknolojileri Ltd.Şti., Bursa, Türkiye
{ sedat.yildiz, erdem.ozgur, nihal.bilal }@teracity.com.tr

Özet. Yüz tanıma teknolojisi kısaca, insan yüzünün karakteristik özelliklerini dijital ortamda analiz ederek biyometrik profil üzerinden kimlik tanılama, kişi doğrulama ve onaylama işlemlerini gerçekleştiren teknolojidir. Yapay zeka teknolojilerindeki gelişmelerle birlikte yüz tanıma sistemleri ile ilgili giderek artmış, bu sistemler %99,83 oranlarında başarım elde edilmiştir[1]. Yüz tanıma sistemlerindeki başarımın artması daha yaygın bir şekilde kullanılmasını da sağlamıştır. Yüz tanıma teknolojisi özellikle güvenlik, suçluların tespiti, ulaşımda hızlı geçişin sağlanması vb. alanlarda yaygın bir şekilde kullanılmaya başlanmıştır. Bu çalışmanın içeriğinde yüz tanıma ile ilgili geliştirilen makine ve derin öğrenme algoritmalarının karşılaştırılması, performans açısından en iyi algoritmanın seçilmesi ve bir yüz tanıma sistemi geliştirilmesi ile ilgili yapılan çalışmalar sunulmuştur. Uygulamada yüz tanıma için 6 farklı algoritma üzerinde çalışılarak bu algoritmaların başarımları ve hızları analiz edilmiştir. Bir sonraki aşamada başarım ve hız açısından en uygun algoritmanın optimizasyonu üzerinde çalışılmıştır. Optimizasyon için algoritmanın testleri için kullanılan veri setleri üzerinde Öklid uzaklık analizi ve Cohen's Distance analizi yapılmış ve en uygun parametreler hesaplanmıştır. Sonraki aşamada bir görüntü işleme servis uygulaması prototipi geliştirilmiştir. Web servis mantığı ile çalışan bu uygulamada haberleşme standartı olarak REST/Json teknikleri kullanılmaktadır. Uygulamada parametre olarak gönderilen bir kare üzerinde yüz tespit ve tanıma yapılmaktadır.

Anahtar Kelimeler: Yüz Tespiti, Yüz Tanıma, Yapay Zeka, Makine Öğrenmesi, Derin Öğrenme

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Development and Optimization of Artificial Intelligence Based Face Recognition System

Sedat Yıldız[1], Erdem Özgür[1], Nihal Bilal[1],

[1]Teracity Yazılım Teknolojileri Ltd.Şti., Bursa, Turkey
{ sedat.yildiz, erdem.ozgur, nihal.bilal }@teracity.com.tr

Abstract. Face recognition technology is the technology that performs the identification, authentication and verification processes through biometric profile by analyzing the characteristics of human face in digital environment. With the advances in artificial intelligence technologies, face recognition systems have gradually increased and these systems have achieved 99.83% performance [1]. Increased performance in facial recognition systems has made it more widely used. Face recognition technology, especially security, detection of criminals, ensuring rapid access to transport and so on. areas have been widely used. In this study, the comparison of the machine and deep learning algorithms developed for face recognition, the selection of the best algorithm in terms of performance and the development of a face recognition system are presented. In practice, 6 different algorithms have been studied for face recognition and their performance and speed have been analyzed. In the next stage, optimization of the most suitable algorithm in terms of performance and speed was studied. For the optimization, Euclidean distance analysis and Cohen's Distance analysis were performed on the data sets used for the tests of the algorithm and the most appropriate parameters were calculated. In the next stage, an image processing service application prototype was developed. In this application which works with web service logic, REST / Json techniques are used as the communication standard. In the application, face detection and recognition is made on a square sent as a parameter.

Keywords: Face Detection, Face Recognition, Artificial Intelligence, Machine learning, Deep learning.

1 Giriş

1970'li yıllardan bu yana üzerinde çeşitli çalışmaların yapıldığı ve çeşitli algoritmaların geliştirdiği bir biyometrik teknik olan yüz tanıma Bilgisayarlara insanın görme sistemine benzer şekilde görme ve tanımlama özelliğinin kazandırılması şeklinde tanımlanabilir. Endüstride Personel Devam Kontrol Sitemi (PDKS), güvenlik, sağlık, üye takip sistemi, suçluların tanınması ve takibi farklı alanda kullanılmaktadır.

Yüz tanımada bugüne kadar pek çok yöntem uygulanmış, birbirinden farklı algoritmalar geliştirilmiştir. Uygulanan yöntemleri geleneksel ve modern yöntemler olarak iki gruba ayrılabilir.

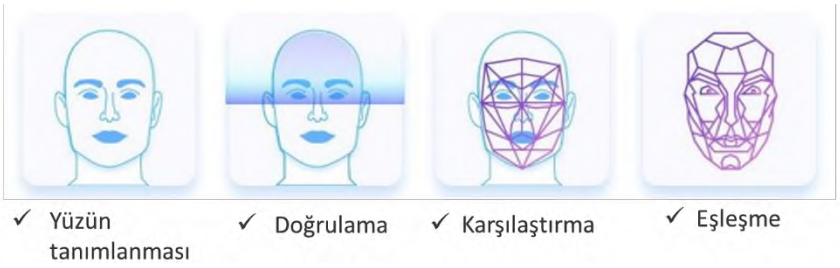
Geleneksel yöntemler tanıma yapılacak yüzü veri tabanındaki yüzlerle karşılaştırarak yüzdeki belirli özelliklere dayalı bir geometrik yaklaşımda bulunur. Sonrasında bu görüntüyü numerik değerlere dönüştürür ve veri tabanındaki görsellerle eşleşip eşleşmediğini bulmak için yüz şablonuyla eşleştirildiği fotometrik bir yöntem kullanır. Fisher vektör algoritması ve Temel bileşenler Analizinde (PCA) kullanılan EigenFaces (Öz yüzler) geleneksel yöntemlere örnek olarak verilebilir. [2]

Bu yöntemler bir görüntüden yüzün özelliklerini çıkarmak için Logic Binary Pattern (LBP) ya da SIFT gibi görüntü tanımlayıcılarını kullanırlar. Modern yüz tanıma yöntemleri ise Makine Öğrenmesi ve derin öğrenme tekniklerine bağlı yöntemlerdir. Bu yöntemler dijital görüntülerle beslendikçe algoritmanın zaman içinde görsellerin özelliklerine bağlı olarak öğrenmesi esasına dayalı bir çalışma prensibine sahiptirler. Bir derin öğrenme algoritması verilerden desenleri veya özellikleri yeterince öğrendikten sonra, daha önce görmediği dijital bir görüntünün özelliklerini çıkarma yeteneğine sahip olabilecek ve bu örnek ya da özellikleri veri tabanında depolanan daha önceki görsellerle kıyaslayarak resmin kime ait olduğunu tanıyabilecektir.

Bu çalışma ile yüz tanıma ile ilgili geliştirilen makine ve derin öğrenme algoritmalarının karşılaştırılması, performans açısından en iyi algoritma seçilmesi ve bir yüz tanıma sistemi geliştirilmesi hedeflenmiştir. İkinci bölümde yüz tanıma hakkında bilgi verilerek karşılaştırılan algoritmaların analizi ve optimizasyonundan bahsedilmiş, sonuç kısmında elde edilen bulgular ve konu ile ilgili muhtemel araştırma konuları belirtilmiştir.

2 Yüz Tanıma

Bir yüz tanıma sisteminin çalışma mantığı aşağıda Şekil 1'de yer alan 4 temel adımda incelenebilmektedir.



Şekil 1. Yüz Tanıma Adımları

Üzerinde çalışılan yüz tanıma algoritmaları iseDlib DNN:ResNet-34 Model [19], Dlib DNN+k-NN :Resnet-29 Model [19], FaceNet 20180402-114759 Model [18], OpenCV LBPH [22], OpenCV Fisherfaces [22], OpenCV Eigenfaces [22], Eigenfaces + k-NN'dir.

2.1 Dlib DNN:ResNet-34 Model

Bu algoritmada Derin metrik öğrenme (deep metric learning) tekniği kullanılarak yüz tanıma işlemi gerçekleştirilmektedir. Metrik öğrenme, yüz örnekleri arasındaki benzerliği veya mesafeyi hesaplamak için eğitim verilerinden bir metrik fonksiyon öğrenir. Ağ mimarisi olarak ResNet-34 [19] modeli kullanılmaktadır. Bu algoritmada her kişi için 128 uzunluklu bir öznitelik vektörü üretilerek yüz tanıma işlemi gerçekleştirilmektedir. Algoritma her kişi için 128 –d vektörlerini elde ettikten sonra Öklid uzaklık hesabıyla yeni bir kişinin hangi sınıfa ait olduğunu belirlemektedir. Kişi tahmini, oylama mantığı ile gerçekleştirilmektedir. Uzaklığın belirli bir eşik değerinin altında olduğu tüm uzaklık değerlerine göre indis değerleri seçilir, daha sonra ilgili indis değeri kullanılarak kişiler oylanır. En çok oyu alan kişi tahmin edilen kişi olarak geri döndürülür.

2.2 Dlib DNN + k-NN : ResNet-29 Model

Bu algoritmada da bir önceki algoritmaya benzer bir şekilde deep metric learning tekniği kullanılmaktadır. Ağ mimarisi olarak ResNet-34 [19] mimarisinin bir versiyonu olan ResNet-29 [19] Modeli kullanılmaktadır. Bu Model kullanılarak her yüz için 128 uzunluklu öznitelik vektörleri elde edilmektedir. Daha sonra bu öznitelik vektörleri ilgili kişi etiketi ile k-NN sınıflandırıcı kullanılarak eğitim ve yüz tanıma işlemi gerçekleştirilir.

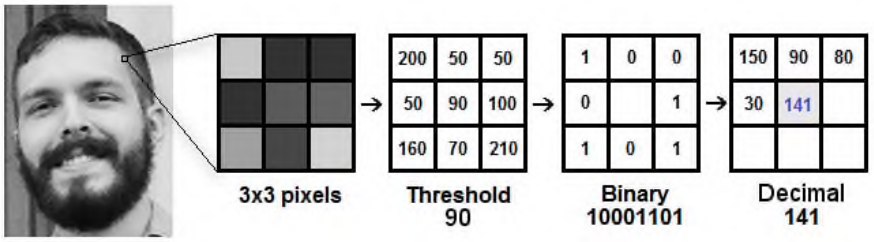
2.3 FaceNet 20180402-114759 Model

FaceNet [18] yüz görüntülerinden doğrudan yüz benzerlik ölçüsüne tekabül eden kompakt bir Öklid uzayına haritalamayı doğrudan öğrenen bir derin öğrenme modelidir. Öklid uzayında mesafeler doğrudan yüz benzerlik ölçüsüne karşılık gelmektedir.[20] Bu algoritma LFW[21] veri seti kullanılarak % 99.65 başarıma sahip, kullanılan modele göre tespit edilip kırpılmış yüzler için 128 veya 512 uzunluklu öznitelik vektörleri üretir. Çalışmada kullanılan model 512 uzunluklu öznitelik vektörleri üretmektedir. Her kişi için öznitelik vektörleri üretildikten sonra k-nn sınıflandırıcı kullanılarak yüz tanıma işlemi gerçekleştirilmektedir.

2.4 OpenCV LBPH

LBP Ojala ve arkadaşları tarafından doku tanımada elde ettikleri başarılı sonuçlar ile ortaya atılmıştır. Bu operatör görüntünün 3×3 pencerelerin ortasındaki piksel değeri eşik seviyesi olarak seçilerek birbirlerine göre karşılaştırılması sonucu ikili değerlerin atanması temeline dayanmaktadır. Oluşturulan ikili sayı dizisine LBP kodu denir ve bu kod ile görüntüdeki farklı tipteki özellikleri belirlemek mümkün

hale gelir. Aşağıda Şekil 2'da LBP kodunun nasıl elde edildiği görsel olarak da belirtilmiştir. LBP kodu elde etme işlemi tüm görsel taranarak tekrarlanır ve her nokta için ayrı ayrı LBP kodu elde edilir. Sonrasında bu kodlardan histogram özellikleri elde edilir. Histogram özellikleri yüz tanımada kullanılır.



Şekil 2. LBP kodunun elde edilmesi

2.5 OpenCV Fisherfaces

FisherFace yüz tanıma algoritması 1997 yılında P. Belhumeur tarafından Lineer Discriminant Analizi (LDA) temel alınarak geliştirilmiştir. Aynı kişiye ait birden fazla farklı görüntünün sınıf içi dağılımının en aza, sınıflar arası dağılımın ise en fazlaya ulaşabilmesi amacıyla geliştirilmiştir[22].

2.6 OpenCV EigenFaces

EigenFace yaklaşımı yüksek boyutlu verilerin Temel bileşenler analizi kullanılarak daha düşük boyutta simgelenebilmesi için , iki boyutlu matrislere uyarlanmış bir uygulamadır. Görsellerin kovaryans matrisinin öz değer ve öz vektörleri bulunur ve yüksek öz değerlere karşılık gelen öz vektörler saklanır. Bu öz vektörler öz yüzlere karşılık gelir. Yeni bir kişi için de öz vektörler hesaplanarak daha önceden kayıt edilmiş özyüzler ile kıyaslanarak yüz tanıma işlemi gerçekleştirilir.

2.7 Yüz Tanımada Kullanılan Veri Setleri

Yüz tanıma algoritmalarının testi için 2 farklı veri setinden faydalanılmıştır. İlk veri seti aşağıda özellikleri maddeler halinde açıklanan Georgia Tech[24] veri setidir.

Georgia Tech veri seti 50 farklı kişinin farklı fotoğraflarından oluşmaktadır. Eğitim için 50 kişinin her biri için ortalama 14'er fotoğrafı, test için de 1'er fotoğrafı kullanılmıştır. Görsellere kırpma ve hizalama (aligning) işlemi uygulanarak eşit boyutlarda ölçeklendirilmiştir. Görseller 350*320 piksel boyutundadır.

Yüz tanıma için kullanılan diğer veri seti ise Teracity Yazılım Teknolojileri personel fotoğraflarından oluşturulmuştur. Bu veri setinde ise Teracity şirket personelinden 27 kişinin ortalama 18 fotoğrafı kullanılarak bir veri seti oluşturulmuş, her kişi için 18 fotoğrafın 15'i eğitim için, 3'ü test için kullanılmıştır. Görsellere kırpma ve hizalama (aligning) işlemi uygulanarak eşit boyutlarda ölçeklendrilmiştir. Görsel Boyutları: 250*350 pikseldir.

2.8 Yüz Tanıma Algoritmaları Analizi

Aşağıda yer alan Tablo 1 ve Tablo 2'de Yüz tanıma algoritmalarının Georgia Tech veri seti ile yapılan testlerinde alınan sonuçları karşılaştırılmıştır. "k" değeri parametresi olarak gösterilen knn sınıflandırma değeri algoritmaların tamamında kullanılmadığı için sonuçlar iki ayrı tablo halinde verilmiştir.

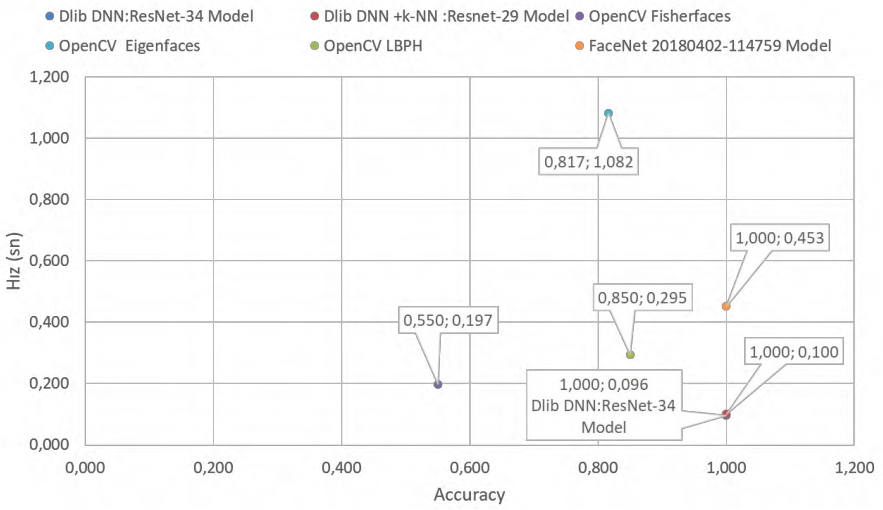
Tablo 1. Yüz tanıma algoritmaları analiz tablosu (Dlib ve Facenet) – GeorgiaTech Veri Seti

Yöntem	Yüz Tespit Yöntemi	k değeri	Eşik Değeri (Threshold)	TP	FP	TN	FN	Ortalama Test Süresi (sn)	Başarım (Accuracy)
Dlib DNN:ResNet-34 Model	Dlib CNN	-	0,45	50	0	10	0	0,0962	1,0000
	Dlib HOG	-	0,45	50	0	10	0	0,2102	1,0000
Dlib DNN +k-NN :Resnet-29 Model	Dlib CNN	5	0,45	50	0	10	0	0,1002	1,0000
	Dlib HOG	5	0,45	50	0	10	0	0,2030	1,0000
FaceNet 20180402-114759 Model	Dlib CNN	5	0,784	50	0	10	0	0,4525	1,0000
	Dlib HOG	5	0,784	50	0	10	0	0,5773	1,0000

Tablo 2. Yüz tanıma algoritmaları analiz tablosu (OpenCV LBPH, Fisherfaces, Eigenfaces, Eigenfaces) – GeorgiaTech Veri Seti

Yöntem	Yüz Tespit Yöntemi	Eşik Değeri (Threshold)	TP	FP	TN	FN	Ortalama Test Süresi (sn)	Başarım (Accuracy)
OpenCV LBPH	Dlib CNN	60	41	7	10	2	0,2952	0,8500
	Dlib HOG	60	46	3	10	1	0,7817	0,9333
OpenCV Fisherfaces	Dlib CNN	1350	24	16	9	11	0,1971	0,5500
	Dlib HOG	1350	40	3	9	7	0,5938	0,8305
OpenCV Eigenfaces	Dlib CNN	8600	39	9	10	2	1,0821	0,8167
	Dlib HOG	8000	40	2	9	8	1,5172	0,8305
Eigenfaces	-	6000000000	45	3	10	2	0,6035	0,9167

Yüz tanıma için incelenen algoritmaların Georgia Tech veri seti üzerinde çalıştırılması ile ilgili elde edilen Accuracy / Hız Grafiği aşağıda Şekil 3'de verilmiştir.



Şekil 3. GeorgiaTech Kişi Veri Seti [24] Başarım (Accuracy) / Hız (sn) Grafiği

Aşağıda yer alan Tablo 3 ve Tablo 4 'te Yüz tanıma algoritmalarının Teracity Personel veri seti ile yapılan testlerde alınan sonuçları karşılaştırılmıştır. Bu veri setinde ki sonuçlar da "k" değeri parametresi olarak gösterilen knn sınıflandırma değeri algoritmaların tamamında kullanılmadığı için iki ayrı tablo halinde verilmiştir.

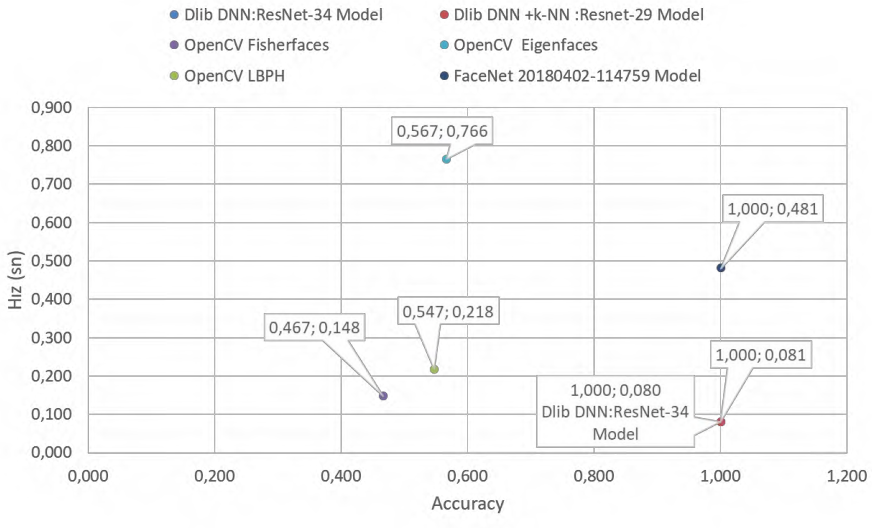
Tablo 3. Yüz tanıma algoritmaları analiz tablosu - Teracity Personel Veri Seti

Yöntem	Yüz Tespit Yöntemi	k değeri	Eşik Değeri (Threshold)	TP	FP	TN	FN	Ortalama Test Süresi (sn)	Başarım (Accuracy)
Dlib DNN:ResNet-34 Model	Dlib CNN	-	0,45	50	0	10	0	0,0962	1,0000
	Dlib HOG	-	0,45	50	0	10	0	0,2102	1,0000
Dlib DNN +k-NN :Resnet-29 Model	Dlib CNN	5	0,45	50	0	10	0	0,1002	1,0000
	Dlib HOG	5	0,45	50	0	10	0	0,2030	1,0000
FaceNet 20180402-114759 Model	Dlib CNN	5	0,784	50	0	10	0	0,4525	1,0000
	Dlib HOG	5	0,784	50	0	10	0	0,5773	1,0000

Tablo 4. Yüz tanıma algoritmaları analiz tablosu - Teracity Personel Veri Seti

Yöntem	Yüz Tespit Yöntemi	Eşik Değeri (Threshold)	TP	FP	TN	FN	Ortalama Test Süresi (sn)	Başarım (Accuracy)
OpenCV LBPH	Dlib CNN	60	41	7	10	2	0,2952	0,8500
	Dlib HOG	60	46	3	10	1	0,7817	0,9333
OpenCV Fisherfaces	Dlib CNN	1350	24	16	9	11	0,1971	0,5500
	Dlib HOG	1350	40	3	9	7	0,5938	0,8305
OpenCV Eigenfaces	Dlib CNN	8600	39	9	10	2	1,0821	0,8167
	Dlib HOG	8000	40	2	9	8	1,5172	0,8305
Eigenfaces	-	6000000000	45	3	10	2	0,6035	0,9167

Teracity Yazılım Personel veri seti üzerinde yüz tanıma algoritmalarının çalıştırılması ile ilgili elde edilen Accuracy / Hız Grafiği ise aşağıda Şekil 4'te verilmiştir.



Şekil 4. Teracity Personel Veri Seti Başarım (Accuracy) / Hız (sn) Grafiği

3. Sonuçlar ve Tartışmalar

Yüz tespit ve tanıma algoritmalarının analiz edilmesiyle elde edilen sonuçlar şu şekildedir:

- ✓ Yapılan çalışmalar sonucunda yüz tespitinde doğruluk oranı en yüksek algoritmalar Dlib HOG ve Dlib CNN çıkmıştır(Şekil 16).

Tablo 5. Yüz tespit algoritmaları en iyi başarım tablosu

	Dataset1	Dataset2
Dlib HOG	0,9965	1
Dlib CNN	0,9931	1

- ✓ Hız açısından en başarılı algoritma Opencv LBP algoritmasıdır ancak çok sayıda FP ve FN mevcuttur.
- ✓ Hem doğruluk hem de hız açısından CPU da çalışma sonucunda en başarılı algoritma Dlib HOG Algoritmasıdır.
- ✓ Dlib CNN algoritması GPU ile çalışmaya uygundur ve GPU ile çalıştırıldığında hızda çok ciddi artış gözlenmekte. Bu da GPU ile birlikte Dlib CNN algoritmasının tercih edilmesinde önemli bir etkendir.
- ✓ En başarılı algoritmalar Dlib Hog ve Dlib CNN algoritmalarıdır ancak Hog frontal face yani ön yüz açısından başarılı iken yan açıdan yüzleri tespit etmekte başarısızdır. Dlib CNN algoritması farklı açılardan yüzleri tespit etmekte oldukça başarılıdır.
- ✓ Her iki veri seti için sonuçlar incelendiğinde başarım ve hız açısından Dlib DNN: ResNet-34 ve Dlib DNN + k-NN :Resnet-29 Modelleri en iyi sonuçlara sahiptir.
- ✓ FaceNet modeli başarım açısından oldukça iyi sonuçlar üretmektedir ancak ResNet modelleri ile kıyaslandığında bu modelde gürültü giderme vb. ön işlemler uygulandığı için yüz tanımada daha yavaş olduğu görülmektedir.
- ✓ ResNet ve FaceNet algoritmaları için cohen's –d analizi verideki değişimi ve en uygun k değerinin ne olduğunu göstermektedir. Cohen's- d grafikleri incelendiğinde ResNet modelinde üretilen değerlerin daha yüksek olduğu görülmektedir. Bu durum ResNet algoritması sonucu üretilen vektörlerin daha iyi ayrıştığını göstermektedir.
- ✓ Klasik algoritmalar olan Eigenfaces ,Fisherfaces ve LBPH algoritmalarının yüz tanıma açısından derin öğrenme algoritmalarına göre başarımı düşüktür .
- ✓ Eigenfaces ve Fisherfaces algoritmalarında kullanılan görseller önişlemden geçirilerek kırpma ve hizalama işlemleri uygulanmaktadır. Bütün resimlerin aynı boyutlarda olmak zorunda olması bu algoritmaların dezavantajlarındandır.
- ✓ Eigenfaces ve Fisherfaces algoritmaları ışık ,yüz açısı gibi etkenlerden çok kolay etkilenmektedir. Bu durum sonuçları olumsuz etkilemekte ve başarımı düşürmektedir. LBPH algoritması ise ışıktan etkilenmemektedir.

- ✓ Derin öğrenme algoritmalarında resim ön işleme (kırpma , hizalama …) gibi işlemlere gerek yoktur ve bu algoritmalar her kişi için özel öznitelik vektörleri çıkardıkları için ışık, açı gibi faktörlerden çok fazla etkilenmemektedir.

Kaynakça

1. M. Wang, W. Deng , Deep Face Recognition: A Survey , arXiv:1804.06655v8 [cs.CV] 12 Feb 2019
2. M. Naeem, I. Qureshi, F. Azam, Face Recognition Techniques And Approaches: A Survey, Sci.Int.(Lahore),27(1),301-305,2015
3. https://www.superdatascience.com/blogs/opencv-face-detection
4. https://github.com/davisking/dlib/blob/master/python_examples/face_detector.py
5. https://github.com/opencv/opencv/tree/master/samples/dnn/face_detector
6. Zhang K. , Zhang Z. , Li Z. , Yu Qiao, Senior , Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks , 2016
7. Viola P. ,Jones M. , Robust Real-time Object Detection , Second Internatıonal Workshop On Statıstıcal And Computatıonal Theorıes Of Vısıon – Modelıng, Learnıng, Computıng, And Samplıng , Vancouver, Canada, July 13, 2001
8. https://github.com/davisking/dlib
9. http://www.image-net.org/
10. https://www.tugraz.at/institute/icg/research/team-bischof/lrs/downloads/aflw/
11. http://host.robots.ox.ac.uk/pascal/VOC/databases.html
12. http://www.robots.ox.ac.uk/~vgg/data/vgg_face/
13. http://shuoyang1213.me/WIDERFACE/
14. http://vintage.winklerbros.net/facescrub.html
15. Shashua A., Gdalyahu Y., and Hayon G., "Pedestrian detection for driving assistance systems: Single-frame classification and system level performance", In Proceedings of IEEE Intelligent Vehicles Symposium, 2004.
16. N. Dalal and B. Triggs., "Histograms of oriented gradients for human detection", In C. Schmid, S. Soatto, and C. Tomasi, editors, International Conference on Computer Vision and Pattern Recognition, volume 2, pages 886–893, June 2005.
17. W. Liu , D. Anguelov , D. Erhan, C. Szegedy , S. Reed , Cheng-Yang Fu , A. C. Berg , SSD: Single Shot MultiBox Detector , arXiv:1512.02325v5 [cs.CV] 29 Dec 2016
18. F. Schroff, D. Kalenichenko, J. Philbin , FaceNet: A Unified Embedding for Face Recognition and Clustering , arXiv:1503.03832v3 [cs.CV] 17 Jun 2015
19. K. He, X. Zhang, S. Ren, j. Sun , Deep Residual Learning for Image Recognition , arXiv:1512.03385v1 [cs.CV] 10 Dec 2015
20. Schroff F. , Kalenichenko D. , Philbin J. , FaceNet: A Unified Embedding for Face Recognition and Clustering, arXiv:1503.03832v3 [cs.CV] ,17 Jun 2015
21. http://vis-www.cs.umass.edu/lfw/
22. https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html
23. C. Szegedy vd. , Going Deeper with Convolutions :GoogleNet , 2015 ,CVPR
24. http://www.anefian.com/research/face_reco.htm

Yazılım Geliştirme Sektöründe Kurumsal Ölçümlerin Belirlenmesi ve Ölçüm Altyapısının Oluşturulması

Melike Turgut[1], Fatma Gül Bulut[1], Haluk Altunel[1]

[1] Softtech A.Ş., İstanbul, Türkiye
{melike.turgut, gul.bulut, haluk.altunel}@softtech.com.tr

Özet. Yazılım geliştirme sektöründe, verimlilik ve üretkenlikle ilgili metriklerinin belirlenmesi ve ölçümlenmesi, işin doğasının soyutluğu sebebiyle, oldukça meşakkatli olabilmektedir. Bunun yanı sıra metriklerin, her sektörde olduğu gibi, organizasyonun ana hedeflerine hizmet edecek şekilde belirlenmesi gerekmektedir. Bunu yapabilmek için de seçilen metriklerin, iyileştirme noktalarının tespit edilmesine yardımcı olması gerekmektedir. Odağında sürekli iyileştirme olan bir şirket için sürekli değişim kaçınılmazdır. Bu nedenle, t anında içinde bulunulan durumun belirlenmesi ve iyileştirme amacıyla alınan aksiyonların t+1'de istenen sonuca doğru götürüp götürmediğinin takibinin yapılması zorunludur. Softtech, Türkiye'de başta bankacılık olmak üzere, yazılım geliştirme sektöründe hizmet vermekte olan bir yazılım şirketi olup, hedefleri arasında farklı sektörlere de hizmet vermek bulunmaktadır. Bu hedefe ulaşabilmek için hem bankacılık sektöründeki ana müşterisinin ihtiyaçlarını karşılamak hem de farklı sektörlerde yeni iş fırsatları yaratmak için verimliliğini ve üretkenliğini arttırmak ihtiyacındadır. Bu amaca uygun olarak, endüstriyel üretimde kullanılan Kanban metrikleri yazılım üretiminde kullanmak üzere uyarlanmış ve Softtech Kurumsal Ölçüm Altyapısı kurulmuştur. Kurulan kurumsal ölçüm altyapısı ile verimlilik ve üretkenliğin sürekli izlenebilmesi sağlanmış, iyileştirme amacıyla alınan aksiyonların etkileri izlenebilir kılınmıştır. Bu bildiride, Softtech'in hedeflerine ulaşmasına hizmet edecek Kanban temelli kurumsal üretkenlik ve verimlilik metriklerinin belirlenmesi, ölçüm altyapısının kurulma süreci, şu ana kadar gerçekleştirilen iyileştirmeler, önümüzdeki dönemde yapılması planlanan diğer çalışmalar aktarılmıştır.

Anahtar kelimeler: Yazılım Geliştirme Metrikleri, Ölçüm, Ölçüm Altyapısı, Verimlilik, Üretkenlik, Tamamlanma Süresi, Gerçekleştirme Süresi, Akış Verimliliği, Kanban, Scrum, Scrumban.

Determination of Corporate Metrics in Software Development Sector and Establishment of Measurement Infrastructure

Melike Turgut[1], Fatma Gül Bulut[1], Haluk Altunel[1]

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

[1] Softtech A.Ş., İstanbul, Türkiye
{melike.turgut, gul.bulut, haluk.altunel}@softtech.com.tr

Abstract. In the software development sector, the identification and measurement of the metrics can be highly engaging due to the abstract nature of the work. In addition, the metrics should be specified to serve the main objectives of the organizations, as in every sector. The selected metrics must assist identifying the points of improvement. For a company whose focus is on continuous improvement, continuous change is inevitable. For this reason, it is inevitable to determine the current situation at time t and follow up whether the actions taken for the purpose of improvement are serving the desired result at a later time point as t+1 or not. Softtech is a software company that serves mainly for the banking sector in Turkey and it aims to provide software services in different sectors as well. In order to achieve this goal, it needs to increase the efficiency and productivity in order to meet the needs of its main customer in the banking sector and to create new business opportunities in different sectors. To this end, Kanban metrics, which are used in industrial production, have been adapted to use in software production and The Softtech Corporate Measurement Infrastructure has been established. With this corporate measurement infrastructure, continuous monirtoring of efficiency and productivity has been achieved, and the effects of actions taken for improvement have became monitorable. In this paper, the identification of the corporate productivity and efficiency metrics to serve the goals of Softtech, the establishment of the measurement infrastructure, the improvements that have been made so far, and the other studies planned to be carried out in the forthcoming period are shared.

Keywords: Software Development Metrics, Measurement, Measurement Infrastructure, Efficiency, Productivity, Lead Time, Cycle Time, Flow Efficiency, Kanban, Scrum, Scrumban.

1 Giriş

Teknolojinin ilerlemesi, bilgiye erişimin kolaylaşması ile birlikte daha da rekabetçi hale gelen endüstri şartları, şirketleri daha verimli çalışmaya zorlamaktadır. Günümüzde pek çok yazılım şirketi rekabette avantaj sağlayabilmek adına çevik ve yalın yazılım yöntemlerine yönelmektedir [1]. Yazılım üretimin hızlanması ve kurumsal çevikliğin artması ile sürekli iyileştirme ve sürekli değişim, şirketlerin normal rutini haline gelmiştir. Buna ek olarak, artan rekabetle birlikte yazılım geliştirme dünyasında proje temelli yaklaşımlar yerini ürün temelli projelere ve modellere bırakmaktadır [2]. Bu değişim, ürünleri en uygun maliyetlerle üretmeyi gerekli kılmaktadır.

Değişimin amacı, içinde bulunulan durumdan daha iyisine ulaşmaktır. İlk kez matematiksel fizikçi ve mühendis Lord Kelvin tarafından kullanılan "ölçemiyorsanız, iyileştiremezsiniz" [3] sözünün de açık bir şekilde vurguladığı gibi değişimi yönetebilmek için değişimin odağındaki hedeflere hizmet edecek metriklerin doğru belirlenmesi ihti-

yacı bulunmaktadır. Amaca hizmet etmeyecek ölçümler, anlamsız veri kümesi olmaktan öteye geçmeyecektir. Doğası soyut kavramlara dayanan yazılım geliştirme sektöründe, beklenen faydaların ölçümlenme formülleri, halen genel standartlar olarak belirlenebilmiş değildir ve çok sayıda yazılım metriği bulunmaktadır [4]. Bu metriklerin önemli bir kısmı kurumsal çevik dönüşüm deneyimlerini merkezlerine almaktadır [5], [6].

Değişimi yönetebilmek için doğru metriklerin belirlenmesi de yeterli olmamaktadır. Metriğin ölçüm yöntemlerinin iyi tanımlanmış olması, ölçüm sonuçlarının objektif çıktılar sağlaması, düzenli bir şekilde gerçekleştirilmesi ve şeffaflıkla sonuçların görünür kılınması gereklidir. Bütün bunların sağlanması durumunda metrikler, yönetim ve çalışanlar tarafından sahiplenilecektir. Unutulmamalıdır ki en objektif, en uygun ve en geçerli metrikler bile sadece birer göstergedir. Metriklerin istenen amaca hizmet etmesi ancak ölçüm sonuçlarının detaylı bir şekilde incelenmesiyle mümkündür. Sonuçların yorumlanabilmesi içinse kullanıcıların, metrikle ilgili tüm açıklamaya sahip olması ve amacına hizmet edeceğine inanması gerekir [7].

Ana hizmet sektörü bankacılık olan Softtech'ten yazının geri kalanında "Firma" olarak bahsedilecektir. Firma, amacını sadece müşterileri için değil aynı zamanda, potansiyellerini gerçekleştirebilmek adına, çalışanları için "Hayata zaman yaratmak." olarak belirlemiştir. Değişimin, verimliliğin ve üretimdeki ilerlemenin ölçülerek yönetilebilmesi için ise 2017 yılı son çeyreğinde Kurumsal Ölçüm Altyapısının temelleri atılmış, 2019 yılı Haziran ayına kadar da kademeli olarak hayata geçirilmiştir. Bu çalışmayla belirlenen Firma kurumsal metrikleri, ölçüm altyapısının oluşturulma süreci, şu ana kadar elde edilen sonuçlar ve olası çalışma konuları aktarılmıştır.

Bildirinin organizasyonu: ilgili çalışmalar ikinci bölümde, hizmet sınıfları, üretim birimleri, kurumsal metrikler üçüncü bölümde, kurumsal ölçüm altyapısının oluşturulması dördüncü bölümde, iyileştirme önerileri, devam eden çalışmalar ve alınan sonuçlar beşinci bölümde, sonuç ve gelecek çalışmalar altıncı bölümde aktarılacak şekilde yapılmıştır.

2 İlgili Çalışmalar

Rekabetçi endüstri şartları, geleneksel yöntemlerle geliştirilmesi çok uzun süren pek çok yazılım geliştirme projesinin daha üretim hattındayken, iş ihtiyaçlarının çok hızlı değişmesi sebebiyle, iptal edilmesine neden olmuştur. 2001 yılında bir araya gelen 17 yazılım gurusu, yazılım geliştirme üretkenliğini ve kalitesini arttırmak için "Çevik Yazılım Geliştirme Manifestosu"nu (Agile Manifesto) oluşturmuştur [8].

Çevik yöntem, bu manifestonun içerdiği 4 değer ve ekinde yayınlanan 12 ilkeye uyan bir grup yazılım geliştirme yöntemini bünyesinde toplamıştır [9].

İlk nesil çevik yöntemlerden olan, yinelemeli ve kademeli bir yaklaşım içeren Scrum popüler olarak seçilen çevik yöntemler arasındadır. Takımların daha verimli çalışması için tasarlanmış Kanban ise Poppendieck'lerin çalışmaları [10] sonucu, Toyota üretim sistemini analizinden sonra Womack tarafından ortaya çıkartılan, "Yalın Düşünme" (Lean Thinking) [11] ve Toyota üretim sisteminin yazılım geliştirme sürecine uyarlan-

masıyla birlikte sonradan bu yöntemlere eklenmiştir. Her iki yöntem de dünyadaki binlerce firma tarafından kullanılmaktadır. Ancak, çevik ekiplerin üyeleri bu yöntemlerden herhangi birinin, her projeye uygulanamayacağı konusunda hemfikirdir [9], [12]. Scrum ve Kanban çevik yazılım geliştirme yöntemleri ile yaşanan zorlukların üstesinden gelmek üzere bu ikisinin karışımı olarak oluşturulmuş Scrumban, her iki yöntemin en iyi yönergelerini bünyesinde barındırır [13]. Çevik yazılım geliştirme projelerinde de ürün temelli yaklaşımlar ön plana çıkmaktadır [14].

Pfleeger'e göre metrikler, ne yaptığımızı ve nasıl yaptığımızı anlamak, kontrol etmek ve iyileştirmek için yaygın olarak kullanılmaktadır [15]. Geleneksel yazılım geliştirme yöntemlerinden çevik yöntemlere geçiş yapan organizasyonlarda, değişimi yönetebilmek için metriklerin kullanımı kaçınılmazdır.

"Uluslararası Standartlaştırma Örgütü" (ISO) ve "Uluslararası Elektroteknik Komisyonu"nun (IEC) yayımladığı ISO/IEC 15939, sistem ve yazılım mühendisliği ve yönetimi disiplinlerine uygulanabilir bir ölçüm sürecini tanımlar. Süreç, hangi ölçüm bilgisinin gerekli olduğunu, ölçümlerin ve analiz sonuçlarının nasıl uygulanacağını ve analiz sonuçlarının geçerli olup olmadığının nasıl belirleneceğini bulmak için gerekli olan ölçüm sürecinin faaliyetlerini adresleyen bir modelle tanımlanmaktadır. Ölçüm işlemi esnektir ve farklı kullanıcıların ihtiyaçlarına göre uyarlanabilir [19].

"Hedef Soru Metriği" (Goal Question Metric - GQM) yaklaşımı ise, organizasyonların amacına uygun bir şekilde ölçmesi için önce kendi hedeflerini belirlemesi gerektiği, ardından bu hedeflere yönelik verilere göre izlemesi gerektiği varsayımına dayanır. Verileri hedeflere göre yorumlamak için bir çerçeve sunmaktadır. Genel anlamda, organizasyonun hangi bilgi gereksinimlerine ihtiyaç duyulduğunu açıkça belirtmek önemlidir; böylece, hedeflere ulaşılıp ulaşılmadığı tespit edilebilir [20].

Yapılan sistematik literatür taramaları, çevik metodolojileri kullanan yazılım organizasyonlarında popüler olan ölçümlerin sprint planlama, iş takibi, yazılım kalitesi, hata çözümleri ve çalışan motivasyonuna odaklanıldığını göstermektedir [3]. Kurgulanacak ölçüm altyapısının başarısının ise kurum tarafından benimsenmesi, "Yazılım Geliştirme Yaşam Döngüsü" (SDLC) ile entegrasyonu, "Zaman Çizelgesi Performans Endeksi" (SPI) ile uyumu ve tasarımıyla ilişkili olduğu düşünülmektedir. [23].

Yazılım geliştirme sektöründe verimlilik ölçümü için belirlenmiş özel bir metrik bulunmamakla birlikte, "Tamamlanma Süresi" (Lead Time) ve bunun alt değerlerinden olan "Gerçekleştirme Süresi" (Cycle Time), çevik yöntem ölçümleri arasında önemli yer teşkil eden metriklerdendir [3], [16]. "Akış Verimliliği" (Flow Efficiency) ise sağladığı diğer faydaların yanı sıra geleneksel "Kaynak Verimliliği"ni (Resource Efficiency) de beraberinde getiren, bu nedenle de takip edilmesi gereken önemli metriktir [17], [18].

Firma tarafından SonarQube kullanılarak "Birim Test Kapsama" (Unit Test Coverage), "Kod Satır Sayısı" (Line of Code) ve "Kural Uyum Endeksi" (Rule Compliance Index), Checkmarx kullanılarak da statik kod analizleri ve güvenlik ölçümleri ayrıca detaylı olarak takip edilmektedir. Verimlilik ve üretkenlik odaklı çevik metodolojiler ayrıca insan faktörünün (motivasyon, yetenek, beceri, iletişim) altını çizmektedir [21]. Firma tarafından insan faktörü üzerine de farklı çalışmalar yürütülmektedir. Bu çalışmada kod güvenlik ve kalitesinden ödün vermeden verimlilik artışı için süre boyutlu metrikler ele alınmıştır.

Hızlanmak ve verimliliğini arttırmak ihtiyacındaki Firma'nın kurumsal metriklerinin belirlenmesi için öncelikle literatür araştırması yapılmıştır. Firma'nın verimliliğinin izlenebilirliği için, ISO/IEC 15939 standartları uygulama rehberi niteliğindeki David N. Card ve J. McGarry'nin çalışmalarından faydalanılarak "Ölçüm Süreç Modeli" (Measurement Process Model), Planlama, Gerçekleştirme, Değerlendirme, Altyapının oluşturulması, baz alınmıştır [22]. Bu süreçte portföy yönetimi, destek hizmetleri ve çevik dönüşümden sorumlu birimlerle birlikte çalışılmıştır.

2017 yılı başında çevik yöntemlere geçiş çalışmalarını başlatan Firma, kurumsal çevik dönüşümünü 2018 yılı sonunda tamamlamıştır. Şirketin büyüklüğü, farklı iş yapış alışkanlıklarına sahip çok çeşitli müşteri yapısı ve yapılan geliştirmelerin doğasının çevikliğe yatkın olmaması nedenleri ile bazı takımların "Şelale" (Waterfall) yöntemiyle geliştirme yapmaya devam etmesine izin verilmiştir. Ölçümlerin bu takımlar için de yapılabilmesi için, bu takımların da bir "İş Listesi" (Backlog) yönetmesi ve Scrumban Tahtası (Scrumban Board) kullanması istenmiştir. Firma'da yukarıdaki ölçümlerinin yanı sıra iyileştirme noktalarını tespit edebilmek adına nedenleriyle birlikte Bloke Süreler ile üretim birimlerinin Bekleme ve Çalışma Sürelerinin ölçülmesi için çalışmalar yapılmıştır.

3 Hizmet Sınıfları, Üretim Birimleri, Kurumsal Metrikler

3.1 Hizmet Sınıfları ve Üretim Birimleri

Firma, yazılım geliştirme hizmetlerinin yanı sıra bakım hizmetleri de sağlamaktadır. Yazılım geliştirme hizmetleri, proje boyutunda olan geliştirmeler ve değişiklik geliştirme taleplerinden oluşmaktadır. Proje niteliğindeki geliştirmeler, teslimatlara kırılarak kademeli olarak gerçekleştirilirken, teslimat ve geliştirme talepleri "Kullanıcı Hikayeleri"ne (User Story) kırılarak yapılmaktadır. Gerçekleştirilen bakımlardan olay ve problem kayıtlarının "Hizmet Seviye Anlaşması" (SLA) bulunmaktadır. Bu üretim birimlerinden teslimat dışındakilerin yönetim ve takibi için Atlassian firmasının Jira proje ve talep yönetimi uygulaması kullanılmaktadır.

3.2 Kurumsal Ana Metriklerin Belirlenmesi

Hızlanma yönünde takip için, çevik yöntem ile geliştirme yapan şirketlerde yaygın olarak kullanılan ve endüstriyel üretimdeki Kanban'dan alınan "Tamamlanma Süresi" (Lead Time) ve "Gerçekleştirme Süresi"nin (Cycle Time) ölçümlenmesine karar verilmiştir. Hesaplama için, hangi üretim birimi için hesaplama yapıldığından bağımsız, tamamlanma süresi gerçekleştirmeye taahhüt verilmesi ile teslim edilme aralığı, gerçekleştirme süresi ise üretime başlanması ile üretimin tamamlanması aralığı olarak tanımlanmıştır.

Tablo 1'de üretim birimleri için ölçümlenen metrikler ile tanımlarına yer verilmiştir.

Tablo 1. Üretim Birimi Tamamlanma ve Gerçekleştirme Süreleri

Üretim Birimi	Metrik	Tanım
Teslimat	Teslimat Pilota Çıkma Süresi	Teslimat çalışmalarının başlaması taahhüt edilen tarih ile pilota çıkması aralığını ifade eder.
	Teslimat Yaygınlaşma Süresi	Teslimat çalışmalarının başlaması taahhüt edilen tarih ile yaygınlaşmanın tamamlanması aralığını ifade eder.
	Teslimat Gerçekleştirme Süresi	Teslimat için çalışmaya başlanması ile teslimatın sürüme hazır olması aralığın ölçümüdür.
Talep	Talep Tamamlama Süresi	Taleplerin planlanması ile üretim ortamına kurulması aralığını ifade eder.
	Talep Gerçekleştirme Süresi	Geliştirmeye başlanması ile kuruluma hazır olması aralığın ölçümüdür.
Kullanıcı Hikayesi	Kullanıcı Hikayesi Gerçekleştirme Süresi	Kullanıcı Hikayesi üzerinde çalışılmaya başlandığı an ile kuruluma hazır hale gelmesi aralığını ifade eder.
	Kullanıcı Hikayesi Tamamlama Süresi	Kullanıcı Hikayesinin İş Listesinde yaratıldığı (yapılmasının taahhüt edildiği) an ile üretim ortamına kurulması aralığını ifade eder.
Olay Kaydı	Olay Kaydı Gerçekleştirme Süresi	Olay Kaydı oluşturulduğu an ile kapatıldığı an aralığını ifade eder.
Problem Kaydı	Problem Kaydı Tamamlanma Süresi	Problem Kaydının yaratıldığı an ile tamamlandığı an aralığını ifade eder.
	Problem Kaydı Gerçekleştirme Süresi	Problem Kaydının üzerinde çalışılmaya başlaması ile sürüme gönderilmesi / tamalanması aralığını ifade eder.
Operasyonel Talep	Operasyonel Talep Tamamlanma Süresi	Operasyonel Talebin Firma'ya iletildiği an ile tamamlandığı an aralığını ifade eder.
	Operasyonel Talep Gerçekleştirme Süresi	Operasyonel Talebin üzerinde çalışılmaya başlaması ile tamalanması aralığını ifade eder.

3.3 Kurumsal Destekleyici Metrikler

İyileştirme noktalarının tespiti ve alınan aksiyonların hedefe yakınlaştırıp yakınlaştırmadığının takibi için kurumsal ana metriklerin yanı sıra bir grup destekleyici metrik belirlenmiştir. Bunlara Tablo 2'de yer verilmiştir.

Tablo 2. Kurumsal Destekleyici Metrikler

Metrik	Tanım
Bloke Süreler	Kullanıcı Hikayeleri üzerinde herhangi bir sebepten çalışılamayan sürelerdir.
Kullanıcı Hikayesi Çalışma Süreleri	Kullanıcı Hikayelerinin çalışma istasyonlarında (analiz, yazılım geliştirme, fonksiyonel test, kullanıcı kabul) geçirdiği sürelerdir.
Kullanıcı Hikayesi Bekleme Süreleri	Kullanıcı Hikayelerinin iki çalışma istasyonu arasında beklediği sürelerdir.
Kullanıcı Hikayesi Akış Verimliliği	Kullanıcı Hikayelerinin ay içinde toplam çalışıldığı sürenin, toplam Çalışma ve Bekleme sürelerine oranıdır.

4 Kurumsal Ölçüm Altyapısının Oluşturulması

4.1 Kurumsal Ölçüm Altyapısı Oluşturma Süreci

Kurumsal Ölçüm Altyapısının oluşturulmasına 2017 yılı sonunda başlanmış ve ölçümlerle ilgili temel çalışmalar 2019 Mayıs sonu itibarıyla tamamlanmıştır.

İlk etapta geliştirme taleplerinin tamamlanma ve gerçekleştirme süreleri ele alınmıştır. geliştirme taleplerinin, teslimatlardan ayrı tutulmasının sebebi, doğası gereği çok daha kısa sürecek taleplerin ölçümlerinin, yine doğası gereği, büyük parçalar olan teslimat ölçümleri ile birlikte yapılması halinde geliştirme taleplerinde tespit edilmesi hedeflenen iyileştirme noktalarının gözden kaçma olasılığı olmuştur.

Çalışmanın başlangıcında, gerçekleştirme ve tamamlanma sürelerinin başlangıç ve bitiş noktaları tespit edilmiştir. Veri kaynağı olan Jira'da, tamamlanma süresi başlangıç noktası olan talebin Firma'ya geldiği an, bitiş noktası olan canlıya kurulma tarihi ve gerçekleştirme süresi başlangıç noktası olan çalışılmaya başlandığı an tespit edilebilirken, süreçte daha önceden tasarlanmadığından, gerçekleştirme süresi bitiş noktası olan kuruluma hazır olma tarihi tutulmadığı için araçta ek geliştirme ihtiyacı doğmuştur.

2018 yılı Şubat ayında, Jira'da gerekli alanın kullanıma açılması ile birlikte hesaplamalara başlanabilmiştir. Hesaplama, Jira'dan talep tarihçesinin alınarak .Net ortamında c# ile yazılan kod ile gerçekleştirilmiştir. Sonuçlarsa şirket geneli / birimler bazında ölçümlerin şeffaf bir şekilde görünür kılındığı, planlamaların, gelişimin, verimliliğin ve risklerin izlenebilirliğini sağlayarak yönetime ve çalışanlara yol gösteren dinamik bir ürün olan "panel.softtech" üzerinden paylaşılmaya başlanmıştır. Kayıt ölçüm detayları ise kurumsal hafızanın tutulduğu Confluence üzerinde sayfalar oluşturularak tüm şirketle paylaşılmıştır.

Firma, stratejik çalışmalarının ve iç fonksiyonlarının sonuçlarını, hedeflerine yönelik endeksler içeren, şirket ve birim "Karne"leri (Balanced Scorecard) ile takip etmektedir. Hesaplamanın hayata geçmesi ve görünür kılınması ile birlikte talep gerçekleştirme ve tamamlanma sürelerinde yapılacak iyileştirme, 2018 yılı şirket ve birim karneleri verimlilik endeksi altında yerini almıştır.

Takibinde, ihtiyaç üzerine, Bloke Sürelerin hesaplanması için çalışmalara başlanmıştır. Öncelikle şirket geneli için geçerli olan toplam 17 farklı bloke nedeni belirlenmiş ve Jira'da kullanıcı hikayelerine bu bloke nedenlerinin girilebilmesi sağlanmıştır. Ardından hesaplama ve gösterim için gerekli yazılım gerçekleştirilmiştir.

Teslimat ölçümleri içinse 3 farklı metrik belirlenmiş, bu metriklerin başlangıç ve bitiş noktaları tespit edilmiştir. Hesaplama bu sefer, Firma teslimat yönetimi için kullanılan altyapıdan tarihçenin alınması ile gerçekleştirilmiştir.

Bu çalışmalar yapılırken yazılım geliştirmenin en küçük yapı taşı olan kullanıcı hikayesine yönelmeye başlanıldığından, kullanıcı hikayesi ölçümlerini objektif yapabilmek adına şirkette Scrumban Tahtası yaygınlaştırılmasına başlanmış, bu çalışma 2018 yılı Kasım ayında tamamlanmıştır. Scrumban Tahtası, ister çevik ister çağlayan yöntem kullansın, tüm üretim takımlarının kullanımına sunulmuştur. Scrumban Tahtası yaygınlaşmanın tamamlanması ile birlikte kullanıcı hikayesi gerçekleştirme ve tamamlanma süreleri hesaplanarak görünür kılınmıştır.

Kullanıcı hikayesi gerçekleştirme ve tamamlanma sürelerinde yapılacak iyileştirme, 2019 yılı şirket ve birim karne hedeflerine girmiştir. Bu nedenle iyileştirme noktalarının bir an önce tespit edilmesine yardımcı olmak adına kullanıcı hikayesi bekleme ve çalışma süreleri ile akış verimliliği hesaplamaları üzerinde çalışılmaya başlanmıştır. 2019 Mart ayında ise hesaplamalar şirketle paylaşılmaya başlanmıştır.

Ölçüm altyapısının son adımları olan olay kayıtları, problem kayıtları ve operasyonel taleplerin ölçümleri ise 2019 yılı Mart ayında tamamlanarak görünür kılınmaya başlanmıştır.

İş Listesi bazında toplam iş miktarı, her iterasyonda tamamlanan iş miktarı, kalan iş miktarı ve eldeki işlerin ne zaman tamamlanacağı öngörüsü konularında görsel destek sağlayan İş Listesi "Yakım Grafiği" (Burndown Chart) da Firma takip panelinde görünür kılınmıştır.

Kullanıcı hikayesi ölçümlerinin tüm takımlar için anlamlı yapılmasını sağlamak adına, kullanıcı hikayesinin doğru işletimini yaygınlaştırmak için bir Kullanıcı Hikayesi Kullanım Kılavuzu oluşturularak Confluence üzerinden tüm şirketle paylaşılmıştır.

4.2 Ölçüm Kural ve Kabulleri

- Ölçümler içinde bulunan karne dönemi için kümülatif hesaplanmaktadır ve her aybaşında bir önceki ayı da kapsayacak şekilde yapılmaktadır.
- Kullanıcı hikayesi ölçümleri sadece Scrumban Tahtalarda akmış kullanıcı hikayeleri için yapılmaktadır.
- Şirket geneli hesaplamasında tüm birimlerin ölçümlerinin ortalamaları kullanılırken, her birim için kendi ölçümlerinin ortalamasından oluşan bir baz değer hesaplanmaktadır. Şirket ve birim karne değerlendirmeleri bu bazlar üzerinde yapılacak iyileştirme üzerinden yapılmaktadır. Hiçbir birim, bir diğeri ile karşılaştırılmamaktadır.
- Tüm süre ölçümleri 24 saatlik iş günü üzerinde yapılmaktadır. Ölçümlere resmi tatiller ve haftasonları dahil edilmemektedir.
- Bloke Süreler, Tamamlanma, Gerçekleştirme, Bekleme ve Çalışma Süreleri ile Akış Verimliliği ölçümlerine dahildir.

4.3 Karşılaşılan Zorluklar

Üçü İstanbul, biri ise Ankara'da olmak kaydı ile 4 farklı konumda yazılım üretimi yapan, farklı iş yapış alışkanlıklarına sahip çok çeşitli müşterilere hizmet veren Firma'nın 19 üretim birimi için her metriğin kurallarının çalışılması gerekmiştir. Bu çalışmalarda üretim birimleri arasındaki farkları dikkate almak gerekmiştir.

Zaman içinde akışlarda yapılan değişiklikler nedeni ile çok büyük boyutlarda olan tarihçe verisinin incelenmesi ve kuralların oluşturulması oldukça meşakkatli olmuştur. Harcanan tüm çabaya rağmen canlıya geçildikten sonra düzeltmeler yapma ihtiyacı doğmuştur.

Daha önceden bu ölçümler olmadığından, akışı zamanında işletilmemiş taleplerin, ölçümleri hatalı sonuç vereceğinden, ortalamaya katılmaması için istisna yönetimi geliştirme ihtiyacı doğmuştur.

Akışları zamanında işletme bilinci daha önce olmadığından birimlerin baz ölçümlerini oluşturmak beklenenden uzun sürebilmiştir.

Veri analizinde, beklenin çok altında süren kayıtlar tespit edilmiştir. Bu kayıtlar incelendiğinde gerekli akışın zamanında işletilmediği fark edilmiştir. Ölçümlendiğinde doğru sonuç vermeyecek bu tür talep ve kullanıcı hikayelerinin tespit edilerek ortalamaya katılmaması için tamamlanma ve gerçekleştirme sürelerine alt limit kontrolü eklenmesi ihtiyacı doğmuştur.

Ölçümlerin karne hedefleri arasına girmesi, bir yandan metriklerin üst yönetim tarafından sahiplenmesi anlamına gelirken, takımların yargılanma kaygısı ile tedirgin olmalarına neden olmuştur. Bu kaygı otomasyonu olmayan akışlarda veri manipülasyonu riskini doğurmaktadır.

Ölçümlerin istenmeyen sonuçları görünür kılabileceği kaygısı taşıyan takımları, Bloke Sürelerin ölçümlere dahil edilmesinin gerekliliği konusunda, ikna etmek gerekmiştir. Ölçümlerin sonuçları kadar "Eğilim"lerinin (Trend) de önemli olduğu, yapılan ölçümlerin sorun noktası tespiti ve alınan aksiyonlar sonrası gelinen durum takibi için gösterge oldukları açıklanmıştır.

Teslimat ile ilgili taleplerin yanlışlıkla talep akışına sokulması nedeni ile burada da istisna yönetimi uygulanması ihtiyacı doğmuş, bu taleplerin Jira projeleri ile ilişkilendirilmeleri sağlanarak geliştirme talebi ölçümlerine dahil olmalarını engellemek gerekmiştir.

Çalışmalara ilk başlandığında paydaş ekiplerle birlikte, hesaplamanın Firma hizmet seviye anlaşması hesaplamasına uygun olarak, 8 saatlik mesai günü üzerinden yapılması tasarlanmış ve bu şekilde hayata geçirilmişken, 2018 yılı içerisinde şirketin esnek çalışma saati uygulamasına geçmesi ile birlikte altyapıda düzeltme yapılması gerekmiş ve iş günü üzerinden hesaplamaya geçilmiştir.

Şirkette yaşanan ürün dönüşümüm çalışmaları ve organizasyon değişiklikleri sebebi ile ölçümlerin doğru birimlerin hanelerine yazılması için ciddi bir operasyon yükü oluşmuştur.

Ölçümlerle ilgili bilgilendirmeler, şirket bünyesindeki tüm araçlar ile (e-posta duyuruları, şirket sosyal medya uygulaması olan Workplace paylaşımları, şirket konumları özelinde yapılan bilgilendirme toplantıları) yapılmaya çalışılsa da iş yoğunluğu nedeni ile tüm takım üyelerine ulaşmayabilmiştir. Bu nedenle tüm birimlerdeki ilgili yönetici

ve takımlara ayrı ayrı ölçüm aktarımları gerçekleştirilmesi gerekmiştir. Yapılan aktarımlarda geri bildirimler alınmış, gerekli aksiyonların alınması sağlanmış, alınabilecek aksiyon bulunmuyorsa da nedenleri ile birlikte gerekli açıklamalar yapılmıştır.

Tüm ölçümlerin takım kırılımında yapılması hedefken sistem kısıtları sebebi ile henüz hesaplamalarda takım kırılımına inilememiştir.

Olay kayıtlarının çokluğu sebebi ile akışın tam zamanında işletilmesi mümkün olmadığında, olay kayıtları için sadece tamamlanma süresi ölçümlenebilmiştir.

Operasyonel talepler ile problem kayıtlarında sorumlu değişikliklerinin çok olması hesaplamayı oldukça karmaşık hale getirmiştir.

Üretimin tamamlanması sonrası müşteri tarafından bir sorun görülmesi halinde Firma'ya iade seçeneği olan üretim birimleri için, tamamlanma ve gerçekleştirme süreleri hesaplamalarından müşteri önünde beklenen sürenin çıkartılması gerekmiştir.

5 İyileştirme Önerileri, Devam Eden Çalışmalar, Alınan Sonuçlar

5.1 İyileştirme Önerileri

Kurumsal metrikler tespit edilerek ölçümlenmiş ve sonuçlar görünür kılınmış olsa da, iyileştirme noktalarının tespitinde büyük iş yine takımlara düşmektedir. Çünkü ancak kayıt detayına inildiğinde iyileştirme noktaları bulunabilecektir. Bu tespitten yola çıkarak "Ölçüm sonuçlarının makine öğrenmesi ile akıllandırılarak, iyileştirme noktaları için takımlara önermelerde bulunulması" iyileştirme önerisi ortaya konulmuştur.

5.2 Devam Eden Çalışmalar

Şirket genelinde devam eden ürün dönüşümünün ölçümler üzerindeki etkisinin analizi yapılarak gerekli değişikliklerin Kurumsal Ölçüm Altyapısına yansıtılmaktadır

Şu ana kadar yapılan ölçümlerden iyileştirme noktalarının tespiti adına ölçüm sonuçlarının anlamlandırılması için veri analitiği çalışmaları gerçekleştirilmekte ve yıl boyunca üretim takımları ile düzenli olarak bir araya gelinerek ölçümlerin değerlendirilmektedir. Bu çalışmalar ile birlikte ölçümlerin devamlılığı sağlanmaktadır.

5.3 Alınan Sonuçlar

2018 karne yılı döneminde (01.02.2018-31.01.2019) üretim birimlerinin araç üzerindeki akışlarının zamanında işletilmesi bilincinin temelleri atılmıştır.

2018 karne yılı döneminde hedeflenen iyileşme 9%'ken şirket talep gerçekleştirme süresinde 28%, talep tamamlanma süresinde 38% iyileştirme gözlemlenmiştir. Bir yıl içinde bu kadar yüksek bir iyileştirmenin sağlanması çarpıcıdır. Takımların ölçümlerin görünür kılınması ile birlikte, ölçüm yapan birimle düzenli iletişime geçerek doğru uygulama konusunda destek istemelerinden anlaşılmaktadır ki, bu iyileşmede, Jira üzerindeki ölçümlerle ilgili verilerin daha dikkatli girilmesinin de payı vardır.

Şirket genelindeki inanç en maliyetli bloke nedeninin müşteri olduğuyken, ölçümlerin yapılması ile birlikte en maliyetli bloke nedeninin bağımlılık yönetiminde yaşanan sıkıntı olduğunun rakamlarla ortaya konmuştur. Bu da hakikat ve algı arasındaki farkın bir göstergesi olmuş ve ölçümlerin gerekliliğini pekiştirmiştir.

Şirket genelinde Akış Verimliliği Kasım 2018'den Nisan 2019 sonuna kadar 6% artmıştır.

6 Sonuç ve Gelecek Çalışmalar

Bu çalışmayla yazılım geliştirme sektöründe, çevik metodolojinin faydalandığı metriklerden bazıları incelenmiş, Firma'nın hedeflerine hizmet etmesi planlanan metrikler nedenleri ile birlikte açıklamıştır. Belirlenen metrik ölçümlerinin standart bir şekilde yapılarak raporlanması, Kurumsal Ölçüm Altyapısının hayata geçirilmesi, tecrübesi aktarılmıştır. Büyük ölçekli bir yazılım geliştirme şirketinde ölçüm altyapısı oluşturma sürecinde dikkate alınması gereken faktörlere değinilerek, karşılaşılan zorluklar paylaşılmıştır.

Yapılan vaka çalışmasında; ölçüm sonuçları, genel inançların gerçekleri yansıtmayabileceğine dikkat çekmiştir.

Bu çalışmayla araç üzerinden toplanan veriler ile yapılan düzenli ölçümler şirket genelinde görünür kılınmıştır. Ölçümlerde gerçekleşen iyileşmenin verimliliğin artışını doğru ifade ettiğine dair henüz yeterli veri bulunmamaktadır. İlerleyen zamanlarda, aynı çalışmanın farklı firmalarda gerçekleştirilmesi halinde sonuçlar hakkında daha net fikir sahibi olunabilecektir.

Sunulan iyileştirme önerisi olan, makine öğrenmesi ile ölçümlerin anlamlandırılarak takımlara önerilerde bulunulması için 2019 yılının ikinci yarısında çalışmalar yapılması planlanmaktadır.

Kaynakça

1. Kiss, F., Rossi, B.: Agile to Lean Software Development Transformation: A Systematic Literature Review. 2018 Federated Conference on Computer Science and Information Systems (FedCSIS) (pp. 969–973). IEEE, (2018).
2. Altunel, H.: Product life cycle based project management model. The Journal of Modern Project Management, 4(3) (2017).
3. Kupiainen, E., Mantyla, M. V., Itkonen, J.: Using metrics in Agile and Lean Software Development - A systematic literature review of industrial studies. Information and Software Technology Journal Volume 62 (pp. 143-163) (2015).
4. Fenton, N. E., Pfleeger S. L.: Software metrics: a rigorous and practical approach. Methods, vol. 5, p. 26 (2009).
5. Heidenberg, J., Weijola, M., Mikkonen, K., Porres I.: A metrics model to measure the impact of an agile transformation in large software development organizations. International Conference on Agile Software Development (pp. 165-179) (2013).

6. Olszewska, M., Heidenberg, J., Weijola, M., Mikkonen, K., Porres I.: Quantitatively measuring a large-scale agile transformation. Journal of Systems and Software, vol. 117 (pp. 258-273) (2016).
7. Klubeck M.: Metrics. How to Improve Key Business Results. Berkeley, CA : Apress, (2011).
8. Beck, et al., "Agile Manifesto", http://www.agilemanifesto.org/, (2001).
9. Alqudah M., Razali R.: A comparison of scrum and Kanban for identifying their selection factors. Proceedings of the 2017 6th International Conference on Electrical Engineering and Informatics: Sustainable Society Through Digital Innovation. ICEEI, (2017).
10. Poppendieck, M., Poppendieck, T.: Lean software development: an agile toolkit. Addison-Wesley Professional, (2003).
11. Womack, J. P., Jones, D. T.: Lean Thinking: banish waste and create wealth in your corporation. Simon and Schuster, (1996).
12. Lei, H., Ganjeizadeh F., Jayachandran P. K., Ozcan P.: A statistical analysis of the effects of Scrum and Kanban on software development projects. Elsevier Robotics and Computer-Integrated Manufacturing Journal Volume 43 (pp. 59-67) (2017).
13. Patil, S. P., Neve, J. R.: Productivity Improvement of Software Development Process Through Scrumban: A Practitioner's Approach. 2018 International Conference On Advances in Communication and Computing Technology (ICACCT) (pp. 314-318). IEEE, (2018).
14. Altunel, H.: Agile Project Management in Product Life Cycle. International Journal of Information Technology Project Management (IJITPM), 8(2) (pp. 50-63) (2017).
15. Fenton, N. E., Pfleeger, S. L.: Software Metrics : A Rigorous and Practical Approach. London : PWS Publishing Company (1997).
16. Joyce, D.: "Kanban For Software Engineering", https:// leanandkanban.files.word-press.com/2009/04/kanban-for-software-engineering-apr-242.pdf, (2009).
17. Modig, N.: The Efficiency Paradox | Niklas Modig | TEDxUmeå [Video dosyası]. Erişim adresi https://www.youtube.com/watch?v=hGJpez7rvc0, (2016).
18. Lombardi, G.:Being Business Agile Focusing on Flow Efficiency: Tale of a Practitioner's Approach. 2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC) (pp. 113-117). IEEE, (2016).
19. ISO/IEC 15939:2007, "Systems and software engineering - Measurement process", International Organization for Standardization.
20. Basili V. R., Caldiera G., Rombach H. D.:The Goal Question Metric Approach. Encyclopedia of Software Engineering, 2nd Edition, Vol. 1, (pp. 578-583) (2002).
21. Highsmith J., Cockburn A.: Agile Software Development: The People Factor. Computer Journal Vol. 34, No 11 (pp 131-133) (2001).
22. McGarry J., Card D., et al., Practical Software Measurement, Addison Wesley (2002).
23. Tahir T., Rasool G., Gencel C.: A systematic literature review on software measurement programs. Information and Software Technology Journal Vol. 73 (pp. 101-121) (2016).

Türkiye Ulusal Yazılım Mühendisliği Sempozyumu Bildirilerinin Konu Analizi

Görkem Giray [0000-0002-7023-9469]

Bağımsız Araştırmacı, İzmir, Türkiye
gorkemgiray@gmail.com

Özet. *Bağlam:* Ulusal Yazılım Mühendisliği Sempozyumunda (UYMS) 2007 ile 2018 tarihleri arasında sunulan bildiriler. *Amaç:* UYMS'de sunulan bildirilere odaklanarak yazılım mühendisliği (YM) araştırma literatürünün yıllar boyunca bir konu analizini yapmak. *Yöntem:* Dünyanın önde gelen YM dergilerinde ve konferanslarında yayımlanan 32.701 yayını kullanarak YM araştırma alanında 11 önemli konu tespit ettik. Bu konu modelini kullanarak 564 UYMS bildirisini analiz ettik. *Sonuçlar:* Yazılım süreci tüm UYMS etkinliklerinde en çok irdelenen konu olmuştur. Son yıllarda uygulamalar konusu da YM araştırmacılarının ilgisini çekmektedir. Yazılım test etme 2013 yılından beri artan bir ilgi görmektedir. Modelleme, tasarım ve mimari ile birlikte kalite ve metrikler son yıllarda en çok ilgi gören beş konu arasındaki diğer iki konudur.

Anahtar Kelimeler: Yazılım Mühendisliği, Araştırma Literatürü, Konu Analizi, Konu Modelleme, Eğilim Analizi.

Topic Analysis of Turkish National Software Engineering Symposium Papers

Görkem Giray [0000-0002-7023-9469]

Independent Researcher, Izmir, Turkey
gorkemgiray@gmail.com

Abstract. *Context:* The papers presented in Turkish National Software Engineering Symposium (NSES) between 2007 and 2018. *Objective:* To conduct a thematic analysis of software engineering (SE) research literature over the years focusing on those papers published in NSES. *Method:* We identified 11 critical topics in SE research using a corpus of 32,701 papers published in leading SE journals and conference proceedings worldwide. We used this topic model to analyze 564 NSES papers. *Results:* Software process has been the top topic in all NSES events. Applications attract SE researchers' attention in last years. Soft-

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

ware testing draw an increasing interest since 2013. Modeling, design and architecture along with quality and metrics are the other two SE topics among the top five in recent years.

Keywords: Software Engineering, Research Literature, Topic Analysis, Topic Modeling, Trend Analysis.

1 Giriş

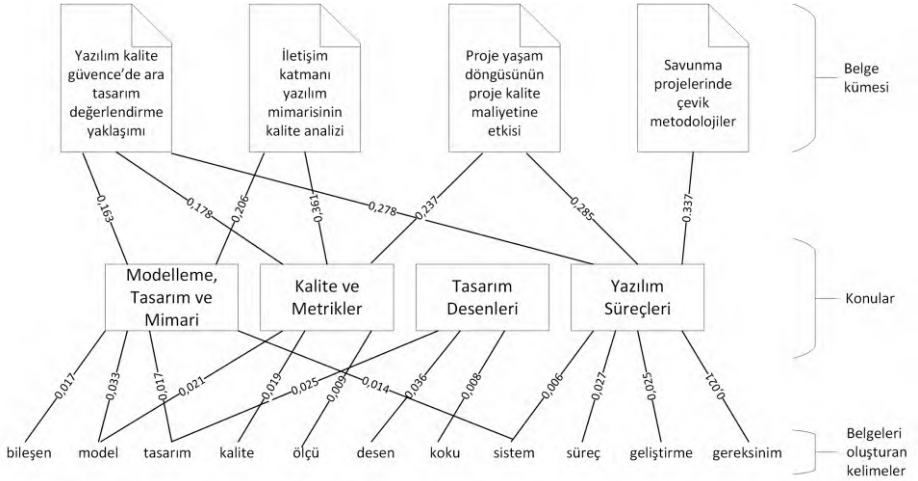
Ulusal Yazılım Mühendisliği Sempozyumu (UYMS) 2003 yılından beri Türkiye'deki yazılım mühendisliği (YM) alanında çalışan araştırmacıları bir araya getirmektedir. Bu sempozyumda sunulan bildirilerin Türkiye'deki YM alanındaki araştırma çalışmalarıyla ilgili genel eğilimleri temsil edebileceği söylenebilir [1]. Bu çalışma kapsamında YM alanında Türkiye'de yapılan araştırma çalışmalarındaki eğilimleri tespit etmek amacıyla UYMS'de 2007 ile 2018 yılları arasında sunulan 564 bildiri konu modelleme yöntemi kullanılarak analiz edilmiştir. Türkiye'de düzenlenen YM alanındaki bir başka konferans olan Ulusal Yazılım Mimarisi Konferansında (UYMK) sunulan bildiriler analize dahil edilmemiştir. Bunun nedeni bu çalışmanın amacının YM alanındaki genel eğilimleri tespit etmek olması ve UYMK'deki bildirilerin ise daha çok yazılım mimarisini irdelemesidir.

2 Konu Modelleme

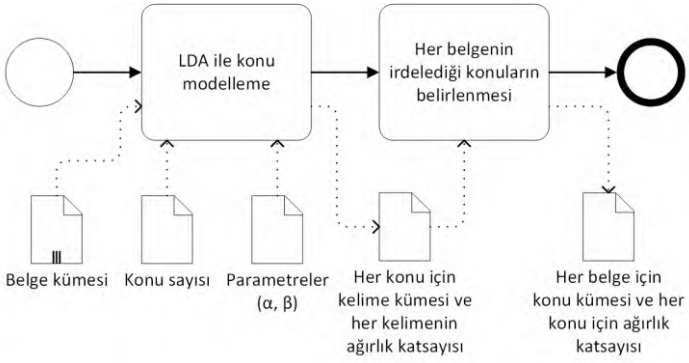
Konu modelleme, farklı sayıda kelimeden oluşan çok sayıda belgenin irdelediği konuları analiz etmeyi amaçlamaktadır [2]. Konu modelleme algoritmaları, belgelerdeki kelimeleri analiz etmek için istatistiksel yöntemleri kullanmaktadır ve önceden etiketlenmiş belgelere (belgelerin hangi konuları içerdiğini belirten bir girdiye) ihtiyaç duymamaktadır [2]. Şekil 1'de bir belge kümesi, konular ve belgeleri oluşturan kelimeler arasındaki ilişki görülmektedir. Belgeleri oluşturan kelimeler toplu olarak istatistiksel yöntemler kullanılarak analiz edilerek belgelerin irdelediği konular keşfedilmektedir. Her bir konu, bir kelime kümesinden ve her kelimenin ağırlık katsayısından oluşmaktadır. Her bir belge de bir ya da birden fazla konuyu değişen ağırlıklarda irdelemektedir.

Konu modelleme için kullanılan değişik yaklaşımlar bulunmaktadır [3], [4]. Latent Dirichlet Allocation (LDA) belge kümelerindeki belgelerin hangi konuları irdelediğini bulmak için kullanılan bir istatistiksel modeldir [4]. Şekil 2'de LDA ile konu modelleme süreci gösterilmektedir. Sürecin girdileri belge kümesi ve bu belge kümesindeki belgelerin irdelediği düşünülen konu sayısıdır. Bazı problemler için konu sayısı önceden bilinebilirken bazı problemler için bu sayı konu modelleme adımının yinelenmesi sonucunda deneysel olarak tespit edilebilmektedir. Oluşturulacak istatistiksel model için gerekli olan α ve β parametreleri ise genellikle çeşitli optimizasyon yöntemleri kullanılarak tespit edilmektedir. LDA ile bir konu modeli oluşturulduktan sonra her bir konu için bir kelime kümesi ve bu kümedeki her bir kelime için de bir ağırlık katsayısı elde edilmektedir. Her bir konu kümesinin temsil ettiği konu başlığı ise kullanıcı tara-

fından belirlenmektedir. Elde edilen konular kullanıcının ihtiyaçlarını karşılayacak şekilde elde edilene kadar ilk adım yinelenebilir. Konular elde edildikten sonra ise her bir belge için o belgenin irdelediği konular ve bu konuları ne kadar irdelediğini belirten katsayılar hesaplanmaktadır.



Şekil 1. Konu modellemede belgeler, konular ve kelimeler arasındaki ilişkiler.



Şekil 2. LDA ile konu modelleme süreci.

LDA, farklı alanlardaki çalışmaları analiz etmek ve zaman içindeki değişimleri görmek için sıkça kullanılmaktadır. Griffiths ve Steyvers, Proceedings of the National Academy of Sciences of the US'deki yayınların içeriklerini, içeriklerdeki zaman içindeki değişimleri ve yayınlar arasındaki benzerlikleri tespit etmek amacıyla LDA kullanmışlardır [5]. Bittermann ve Fischer, PSYNDEX veritabanında bulunan, 1980 ile 2016 yılları arasında psikoloji alanında Almanca ve İngilizce dillerinde yayımlanmış çalışmaları LDA ile analiz ederek bu alandaki eğilimleri raporlamışlardır [6]. Kim ve Delen, tıbbi bilişim alanındaki ana konuları ve bu konulardaki zaman içinde değişimleri LDA kullanarak tespit etmişlerdir [7]. Hall ve arkadaşları, hesaplamalı dilbilim alanında 1978

ile 2006 yılları arasında yapılmış çalışmaları LDA kullanarak analiz etmişler ve bu alandaki eğilimleri zamana bağlı değişimleriyle birlikte raporlamışlardır [8]. YM alanında yapılan buna benzer çalışmalar bir sonraki bölümde sunulmaktadır.

3 İlgili Çalışmalar

YM araştırma alanındaki eğilimleri ve zaman içindeki değişimleri analiz etmek için yapılan çalışmalardan bazıları Tablo 1'de gösterilmektedir.

Tablo 1. YM alanındaki ilgili çalışmalar

Ka.	Yıl	Kapsam	Sınıflandırma Yöntemi
[9]	2002	YM alanında önde gelen altı dergide 1995 – 1999 arasında yayımlanmış 369 makale	İki araştırmacı iki seviyeli bir sınıflandırma şeması oluşturarak yayınları bu şemaya göre sınıflandırmıştır.
[10]	2008	2006 yılında YM alanındaki yedi dergide yayımlanan ve yedi konferansta sunulan toplam 691 yayın	Yayınlar ACM Computing sınıflandırma sistemine göre yazarlar tarafından sınıflandırılmıştır.
[11]	2011	2003 – 2010 arasında UYMS, UYMK ve YKGS'de sunulan 290 bildiri	Yayınlar iki araştırmacı tarafından SWEBOK'taki YM alanları ve yazarlar tarafından bu alanlara eklenen üç alana göre sınıflandırılmıştır.
[12]	2014	2008 ile 2013 yılları arasında 7.638 yayın	Makro seviyede EI Compendex sınıflandırma sistemine ve mikro seviyede yazarların belirledikleri alanlara göre yayınlar sınıflandırılmıştır.
[13]	2015	Empirical Software Engineering and Measurement (ESEM) konferansında 2007 ile 2014 yılları arasında sunulmuş ve Scopus veritabanından elde edilmiş 513 yayın	Yayınların başlıkları ve özetleri LDA kullanılarak analiz edilmiştir.
[14]	2015	2003 – 2014 arasında 8 UYMS'de sunulan 356 bildiri	Yayınların her biri en az iki araştırmacı tarafından birbirinden bağımsız olarak sınıflandırılmıştır.
[1]	2015	Scopus veritabanından elde edilen, başlığında "software" kelimesi bulunan ve yazarların en az birinin mensup olduğu kurumun Türkiye olduğu toplam 289 yayın	Yayın başlıklarındaki terimlerin kullanım sıklıkları analiz edilmiştir.
[15]	2016	2014 yılının sonuna kadar yayımlanmış ve Scopus veritabanından elde edilen 71.668 yayın	Yayınların başlıkları ve özetleri LDA kullanılarak analiz edilmiştir.
[16]	2018	YM alanındaki dergilerde ve konferans kitaplarında son 25 yılda basılan 35.391 yayın	Yayınların başlıkları ve özetleri LDA kullanılarak analiz edilmiştir. Yayınlar 10 başlık altında sınıflandırılmıştır.

2011 ve 2015 yıllarında Türkiye'de düzenlenen YM konferanslarında sunulan bildirileri sınıflandıran iki çalışma yapılmıştır [11], [14]. 2011'de yapılan çalışmada üç konferansta sunulan 290 bildiri yazarlar tarafından SWEBOK YM alanları ve yazarlar tarafından bunlara eklenen üç YM alanına göre sınıflandırılmıştır [11]. 2015'te yapılan çalışmada ise 2003 ve 2014 yılları arasında UYMS'de sunulan 356 bildiri yine yazarlar tarafından konularına göre sınıflandırılmıştır. Yurt dışında gerçekleşen Empirical Software Engineering and Measurement konferansında 2007 ve 2014 yılları arasında sunulan 513 bildiri LDA kullanılarak sınıflandırılmıştır [13]. 2015 yılında yapılan başka bir çalışmada Scopus veritabanında bulunan ve Türkiye'de bir kurumdan en az bir yazar içeren 289 yayın başlıklardaki kelimelerin sıklıkları aracılığıyla analiz edilerek Türkiye'de YM araştırma alanındaki eğilimler belirlenmiştir [1]. 2016 ve 2018 yıllarında yapılan iki çalışmada, daha önce yapılan çalışmalara kıyasla çok daha fazla yayın LDA ile analiz edilerek sınıflandırılmıştır [15], [16].

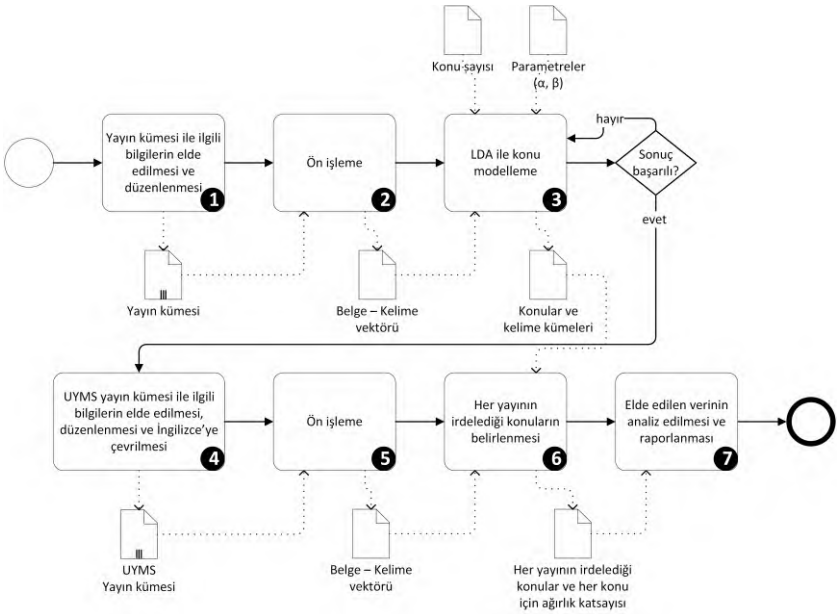
Tablo 1'de görüldüğü gibi 2015 yılından itibaren yayın sınıflandırma için konu modelleme yaklaşımları ve özellikle LDA kullanılmaya başlanmıştır. Böylece birkaç yüz ya da bin yayının yazarlar tarafından manuel bir yöntemle analiz edildiği çalışmaların yanında on binlerce yayının analiz edildiği çalışmalar yapılmaya başlanmıştır.

Bu çalışma kapsamında UYMS'de 2007 ve 2018 yılları arasında sunulan 564 bildiri LDA konu modelleme yaklaşımı kullanılarak sınıflandırılmıştır. Bu sınıflandırma işlemi için kullanılan konu modeli ise [16] çalışmasındaki yayın havuzu kullanılarak oluşturulmuştur. Böylece YM alanındaki konuların dünya çapında tespit edilmesi hedeflenmiştir. Araştırma yöntemi ile ilgili detaylar bir sonraki bölümde anlatılmaktadır.

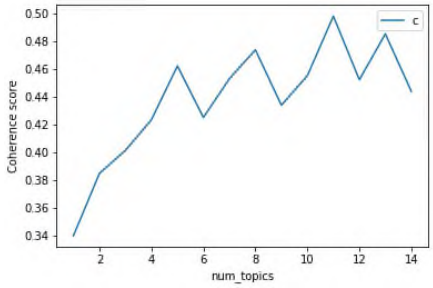
4 Araştırma Yöntemi

Şekil 3 bu araştırmada izlenen sürecin yedi ana adımını ve her adımın girdilerini ve çıktılarını göstermektedir. Birinci adımda, [16] çalışması kapsamında oluşturulan 32.701 yayından oluşan ([16] makalesinde bu sayı 35.391 olarak raporlanmış olsa da makale içerisinde sunulan veri kaynağında 32.701 adet yayın için bilgi bulunmaktadır) veri (yayın yılı, yayın başlığı, yayın özeti gibi) elde edilmiş ve bir Excel dosyasına kaydedilmiştir. [16] çalışmasındaki konu modelinin elde edilmesi amaçlandığı için veri içeriğinde herhangi bir değişiklik yapılmamıştır. İkinci adımda, her bir yayının başlığı ve özeti birer kelime vektörüne dönüştürülmüştür ve bu kelime vektörlerinden etkisiz kelimeler (stop words) ayıklanmıştır. Her bir yayın, veri kümesinde var ise o yayının özetinin kelime vektörü, yoksa o yayının başlığının kelime vektörü tarafından temsil edilecek şekilde düzenlenmiştir. Elde edilen bu kelime vektörleri üçüncü adımdaki LDA ile konu modelleme adımının girdisi olarak kullanılmıştır. Bu adımın diğer girdileri ise konu sayısı ile α ve β parametreleridir. Bu girdiler [16] çalışmasında belirtildiği gibi sırasıyla "11", "0,847" ve "0,764" olarak alınmıştır. Bu parametreler eldeki veri kümesine ve amaca göre (eldeki belgeleri kaç farklı konuya göre sınıflandırmak istendiği) değişmektedir. Bu çalışmada, [16] çalışmasındaki veri kümesi aynı amaçla kullanıldığı için konu modelleme için de aynı parametreler kullanılmıştır. Konu sayısının optimum olup olmadığını kontrol etmek için gensim [17] kütüphanesindeki optimum

konu sayısını [18] çalışmasına göre belirleyen işlevler kullanılmıştır. Şekil 4'teki grafikte görüldüğü gibi 11 konu sayısı eldeki veri kümesinde en yüksek bağdaşıklık (coherence) değerini vermektedir.



Şekil 3. Araştırma sürecinin ana adımları ve her adımın girdileri ve çıktıları.



Şekil 4. Konu sayısı ile elde edilen konuların bağdaşıklıkları (coherence) arasındaki ilişki.

Üçüncü adımdaki işlemler sonucunda elde edilen konular ve bu konuları temsil eden en önemli on kelime önem sırasına göre Tablo 2'de gösterilmektedir. [16] çalışmasında verilen detaylar doğrultusunda bu çalışmadaki konu modeli birebir elde edilememiştir. Tablo 2'de kalın ve eğik yazılmış konular ve kelimeler [16] çalışmasındaki konu modelindeki konular ve kelimelerle ortaktır. Buna göre birebir olmasa da [16] çalışmasına yakın bir konu modeli elde edilmiştir. [16] çalışmasında bulunmayan konular (yazılım ürün hattı, eğitim, eski yazılım, tasarım desenleri) ise YM alanındaki çalışmaları ayırt etmek için kullanılabilecek başlıklar olarak göze çarpmaktadır.

Tablo 2. Konu modelindeki 11 konu ve her konunun en önemli 10 kelimesi (en önemliden daha az önemliye doğru). Kalın ve eğik yazılmış konular ve kelimeler [16] çalışmasındaki konu modelindeki konular ve kelimelerle ortaktır.

Konu	Kelimeler
Yazılım Ürün Hattı	product, feature, line, reuse, configuration, variability, family, variant, spl, asset
Test Etme	***test***, ***testing***, ***case***, ***fault***, ***technique***, ***generation***, suite, ***coverage***, failure, automated
Modelleme, Tasarım ve Mimari	***model***, ***component***, ***design***, ***language***, ***system***, ***specification***, approach, ***architecture***, ***object***, tool
Kalite ve Metrikler	model, ***quality***, method, ***data***, ***metric***, ***defect***, measure, empirical, cost, ***effort***
Eğitim	student, programming, learning, risk, course, university, education, game, note, public
Yazılım Süreci	***process***, ***development***, ***requirement***, project, practice, management, experience, ***system***, goal, organization
Eski Yazılım	reverse, legacy, object_oriented, workflow, reengineering, migration, real_time, ada, cobol, debt
Kaynak Kod	***code***, ***source***, change, ***tool***, ***developer***, ***information***, bug, approach, open, evolution
Uygulamalar	***application***, ***service***, ***web***, ***distributed***, ***system***, security, network, performance, time, ***user***
Tasarım Desenleri	pattern, review, design, detection, clone, smell, bad, anomaly, systematic, recognition
Program Analizi	***program***, ***analysis***, algorithm, technique, ***execution***, state, time, problem, ***dynamic***, data

Dördüncü adımda UYMS'de 2007 ile 2018 yılları arasında sunulan bildirilerin başlıkları, özetleri ve anahtar kelimeleri bir Excel dosyasında birleştirilmiştir. 2003 ve 2005 tarihlerinde düzenlenen UYMS bildirilerine toplu olarak ulaşılamadığı için bu bildiriler kapsam dışında bırakılmıştır. Elde edilen 564 yayına ait başlık, özet ve anahtar kelimeler Google Translate servisi kullanılarak İngilizce'ye çevrilmiştir. Beşinci adımda her bir yayın için kelime vektörleri oluşturulmuştur. Altıncı adımda, her yayının irdelediği konular, üçüncü adımda elde edilen konu modeli kullanılarak belirlenmiştir. Yayınların İngilizce'ye çevrilmiş özetleri konu modelinin girdisi olarak kullanıldığında yayınların irdelediği konuların yeterince ayrıştırılamadığı görülmüştür. Bunun nedeni özet metinlerinin İngilizce'ye çevrilmesinin başlıklara ve anahtar kelimelere göre daha zor olmasından ve çeviri kalitesinin düşük olmasından kaynaklandığı düşünülmektedir. Bundan dolayı konu modelleme girdisi olarak sadece başlık bilgileri ve varsa anahtar kelimeler kullanılmıştır. Diğer çalışmalarda da özetlerin olmadığı durumlarda sadece yayın başlıklarının kullanıldığı görülmektedir [1], [16]. Tablo 3'te örnek olarak beş yayının irdelediği konular ve bu konuların ağırlıkları gösterilmektedir. Yayın no her yayının biricik numarasını göstermektedir. Tablo 3'te görüldüğü gibi bir yayın bir ya da birden çok konuyu değişen ağırlıklarda irdeleyebilmektedir. Bu çalışmada konu katsayısını analize dahil etmek için eşik değeri 0,15 olarak belirlenmiştir. Bu konuda literatürde

bir altın standart bulunmamaktadır. Sonuç olarak her yayın ortalama 1,7 konuyu irdelemektedir.

Tablo 3. Örnek beş yayının irdelediği konular ve bu konuların ağırlıkları.

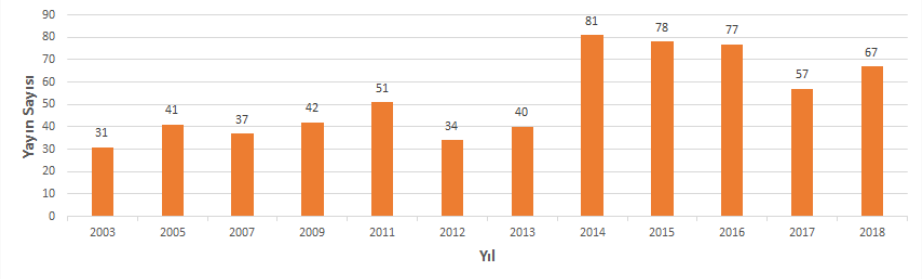
Yayın no	Konular ve ağırlıkları
34	Yazılım süreci: 0,436; Test Etme: 0,182
145	Uygulamalar: 0.537; Kalite ve Metrikler: 0,150
204	Uygulamalar: 0,659
211	Yazılım Süreci: 0,498
379	Test Etme: 0,310; Yazılım süreci: 0,198; Kalite ve Metrikler: 0,185

Yedinci adımda ise elde edilen sonuçlar analiz edilmiş ve bir sonraki bölümde anlatıldığı şekilde raporlanmıştır.

Birinci ve dördüncü adımlarda, ön işleme için hazır hale getirilmesi için veri Excel dosyasında düzenlenmiştir. Yedinci adımdaki analiz için de Excel kullanılmıştır. İkinci, üçüncü, beşinci ve altıncı adımdaki tüm işlemler Jupyter notebook geliştirme ortamında Python programlama dili kullanılarak yazılan prosedürler ile gerçekleştirilmiştir. Veri işleme ve LDA ile konu modellerinin oluşturulması için pandas, gensim, nltk ve lda kütüphaneleri kullanılmıştır.

5 Analiz Sonuçları ve Tartışma

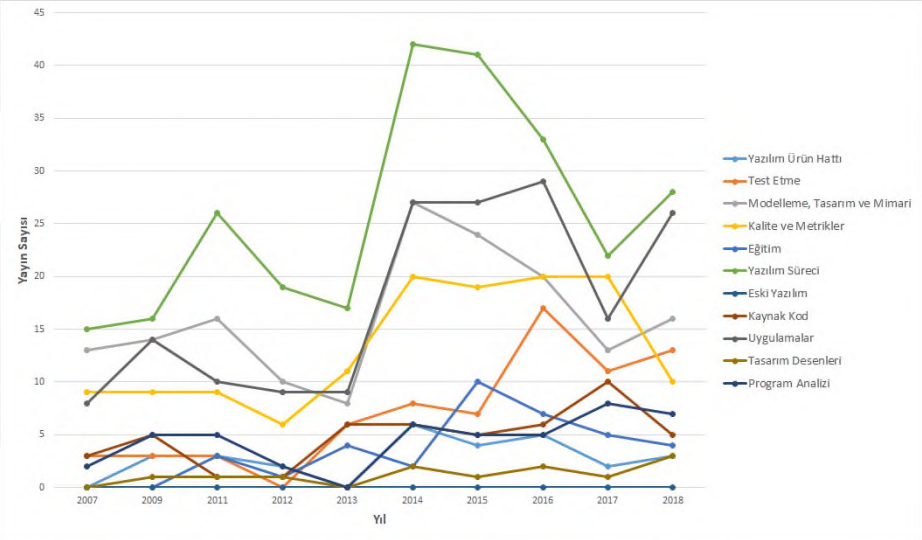
Şekil 5'te UYMS'de sunulan bildirilerin yıllara göre dağılımı gösterilmektedir. 2003 ve 2005 yıllarındaki bildiri sayısı [11] çalışmasından alınmıştır. 2014 yılında konferansta sunulan bildiri sayısında bir sıçrama olduğu görülmektedir. 2014 ile 2018 yılları arasındaki UYMS etkinliklerinde ortalama 72 adet bildiri sunulmuştur.



Şekil 5. Yıllara göre UYMS'de sunulan yayın sayısı.

Şekil 6'da UYMS yayınlarında irdelenen konuların yıllara göre göre değişimi gösterilmektedir. Bir yayın birden fazla konuyu irdeleyebildiği için (Bkz. Bölüm 4 ve Tablo 3) bu grafikteki toplam yayın sayısı 2007 ile 2018 yılları arasındaki yayın sayısından (564 yayın) fazladır. Grafikte en dikkat çekici husus 2014 yılındaki sıçramadır. Bu sıçrama en popüler dört konuda diğer konulara göre daha fazladır. Şekil 5'te görüldüğü gibi

2014 yılındaki toplam yayın sayısındaki ciddi artış bu sıçramayı genel olarak açıklamaktadır.



Şekil 6. UYMS yayınlarında irdelenen YM konularının zamana göre değişimi.

Bu çalışma kapsamında oluşturulan konu modeline göre yazılım süreci konusu her zaman en çok irdelenen konu olarak öne çıkmaktadır. [14] çalışmasında yazarlar tarafından yapılan sınıflandırmaya göre de yazılım süreci en çok irdelenen konulardan biridir. [14] çalışmasında ACM yazılım kategorisi altındaki alt alanlara göre yapılan sınıflandırmada "yazılım geliştirme süreç yönetimi" birinci sırada, SWEBOK konu alanlarına göre yapılan sınıflandırmada "YM süreci" üçüncü sırada yer almaktadır. Diğer taraftan, [16] çalışmasında konferanslar için yapılan sınıflandırmada yazılım sürecinin önemi 2000'li yılların başından itibaren azalma eğilimindedir. Yine aynı çalışmada dergiler için yapılan sınıflandırmada yazılım süreci tüm yıllarda en önemli üç konu arasındaki yerini korumaktadır. [1], [13] ve [15] çalışmalarında ise yazılım süreci en çok irdelenen ilk beş konu arasında bulunmamaktadır.

Yazılım sürecinden sonra en çok irdelenen konu uygulamalar olarak görülmektedir. [16] çalışmasına göre konferanslarda irdelenen konular arasında uygulamalar son on yılda genelde son sıralarda bulunmaktadır. 2015 yılında beşinci, 2016 yılında ise yedinci sırada bulunmaktadır. Dergilerdeki makalelerde ise uygulamalar konferanslara göre daha fazla irdelenmiş olsa da 2015 ve 2016 yıllarında önem açısından sırasıyla beşinci ve yedinci sıralarda bulunmaktadır. [15] çalışmasında 2013 yılının sonuna kadar yapılan çalışmalar için belirlenen en popüler beş konunun ilk sırasında web servisleri ile mobil ve bulut bilişim bulunmaktadır. Bu başlıklar, bu çalışma kapsamındaki uygulamalar konusuna yakın görülebilir (Tablo 2'deki kelimelere göre).

Modelleme, tasarım ve mimari konusu, 2014 yılında önceki yıllara kıyasla UYMS'de ciddi biçimde daha fazla irdelenmiş olsa da 2015 yılından sonra bu konuya ilgi de bir azalma eğilimi görülmektedir. [16] çalışmasında modelleme ve mimari iki

ayrı konu olarak ele alınmıştır. Modellemeyi irdeleyen çalışmaların sayısı konferanslarda son birkaç yılda gözle görülür biçimde azalmaktadır [16]. Konferanslarda mimari konusuna olan ilgi ise yaklaşık son on beş yılda son sıralarda yer almaktadır [16]. Modelleme konusuna olan ilgi, dergi makalelerinde konferanslara oranla daha yavaş olsa da yine de azalma eğilimindedir [16]. Mimari konusuna dergi makalelerinde olan ilgi ise genelde yok denecek kadar azdır [16]. [1] çalışmasında Türkiye'deki araştırmacıların yazılım mimarisine olan ilgilerinin az olduğu raporlanmıştır.

Kalite ve metriklere UYMS katılımcılarının ilgisi 2013 ve 2014 yıllarında artmış sonrasında sabit kalıp 2018 yılında azalmıştır. [16] çalışmasına göre metrikler konferanslarda çok fazla ilgi görmezken dergi makalelerinde metriklere olan ilgide son yıllarda bir miktar artış gözlemlenmektedir. Kalite ve metrikler konusunu oluşturan hata, maliyet efor gibi kelimelere (Tablo 2) bakıldığında, [13] çalışmasında hata ve maliyet kestirimi, [1] çalışmasında hata analizi 2014 yılında en popüler konular olarak göze çarpmaktadır. [15] çalışmasına göre 2013 yılının sonuna kadar en çok yayın yapılan konulardan biri yazılım efor kestirimi olarak görülmektedir.

2013 yılından başlayarak UYMS'de yazılım test etme konusuna artan bir ilgi gözlemlenmektedir. Bununla paralel olarak 2014 yılından itibaren her yıl UYMS kapsamında yazılım test mühendisliği alt alanı kapsamında bildiri çağrısı yapılmaktadır. [16] çalışmasına göre de test etme konusuna konferanslarda olan ilgi son yıllarda ciddi biçimde artmaktadır. Yine aynı çalışmada ise dergi makalelerinde test etme konusuna ilgili genel olarak azdır [16]. [13], [1] ve [15] çalışmalarının hepsinde test etme konusu en popüler konular arasında raporlanmaktadır.

Program analizi ve kaynak kod konularına UYMS katılımcılarının ilgisi son yıllarda çok az artma eğiliminde olsa da bu konulara olan ilgi ilk beş konuya göre oldukça düşüktür. [16] çalışmasının en önemli bulgularından birisi ise YM konferanslarında en çok irdelenen konuların başında yaklaşık son 10 yıldır kaynak kod ve program analizi gelmektedir. Bu bulgunun tersine, aynı çalışmaya göre dergi makalelerinde bu iki konuya olan ilgi oldukça düşük seviyelerdedir [16]. [15] çalışmasına göre kaynak kod konusu en popüler beş konudan birisi olarak karşımıza çıkmaktadır.

Yazılım ürün hattı, eğitim ve tasarım desenleri konularına UYMS'de olan ilgi genelde düşük seviyelerdedir. Yazılım ürün hattı ve eğitim konuları ilgili çalışmalarda da herhangi bir listede yer almamaktadır. Tasarım desenleri ise [15] çalışmasında üzerine en çok yayın üretilen dört konudan biri olarak raporlanmıştır.

UYMS'de eski yazılım (legacy software) konusu hiç irdelenmemiştir. Benzer şekilde [1] çalışmasında da yazılım bakımı ve evrimi Türkiye'deki araştırmacıların en az odaklandığı konulardan birisi olarak raporlanmıştır. Mevcut yazılım sistemlerinin sayısının arttığını, bu sistemlerin artık eskidiğini ve teknolojinin çok hızlı ilerlediğini göz önüne aldığımızda bu konuya hiç ilginin olmaması üzerine düşünülebilir.

Oluşturulan konu modelinde gereksinim mühendisliği ile ilgili bir başlık bulunmamaktadır. [15] çalışmasına göre YM alanında en çok yayın üretilen konuların başında gereksinim mühendisliği gelmektedir. Dolayısıyla UYMS'de gereksinim mühendisliği üzerine çalışmaların teşvik edilmesi düşünülebilir.

Yazılım sistemlerinin bulut altyapısında barındırıldığı ve elektronik verinin hacminin çok arttığı bir ortamda performans ve güvenlik çok önemli iki konu olarak karşımıza çıkmaktadır. [16] çalışmasındaki konu modelinde performans on konudan biri

olarak yer alırken güvenlik bir konu başlığı olarak yer almamıştır. Bu çalışma kapsamında oluşturulan modelde de performans ve güvenlik kelimeleri uygulamalar başlığı altında yer almaktadır. [16] çalışmasına göre performans dergi makalelerinde az da olsa irdelense de konferanslarda hemen hemen hiç irdelenmemektedir. Güvenlik ise ayrı bir konu olarak da ele alınmamaktadır. Benzer şekilde diğer ilgili çalışmaların hiçbirisinde bu iki konu irdelenen bir konu olarak karşımıza çıkmamaktadır. Bu konulardaki ihtiyaçlar düşünüldüğünde konferanslarda ve dergilerde bu konuların irdelenebileceği bir ortamın oluşturulması düşünülebilir.

5.1 Kısıtlar ve Geçerliliğe Yönelik Tehditler

Bu çalışma sadece UYMS'de 2007 ile 2018 yılları arasında sunulan bildirileri kapsamaktadır. Bu yıllar arasındaki tüm bildiriler analize dahil edilmiştir.

Konu modelleme istatistiğe dayanan bir yöntemle (LDA) yapılmıştır. Elde edilen modelin YM alanını iyi temsil edebilmesi için [16] çalışmasındaki 32.701 yayın bilgisi kullanılmıştır. Bu veri kümesindeki eksiklikler ve hatalar bu çalışmada kullanılan model için bir miktar risk oluşturabilir. Bunun yanında YM alanındaki çalışmaların kaç konu altında sınıflandırılabileceği konusunda bir kural yoktur ve olması beklenemez. Bu çalışma kapsamında [16] çalışmasındaki konu sayısı bir girdi alınarak konu modeli oluşturulmuştur. Şekil 4'te gösterildiği gibi 11 konu sayısının 1 ile 15 arasında en iyi sonucu verme olasılığı olan modeldir. Yine de farklı konu sayıları ile farklı modeller elde edilmesi mümkündür. Benzer şekilde farklı α ve β parametreleri kullanılarak farklı modeller oluşturulabilir. Bu parametreler için de [16] çalışmasında optimize edilmiş değerler aynı veri kümesi ile kullanılmıştır.

Konu modelinde elde edilen konular yazar tarafından isimlendirilmiştir. Bu isimlendirme [16] çalışması temel alınarak yapılmıştır. Yine de bu konuların belirlenmesi ve isimlendirmesi sırasında bir taraflılıktan söz etmek mümkündür.

Elde edilen konuların hepsi aynı detay seviyesinde değildir. Yazılım süreci daha genel bir alanı temsil ederken tasarım desenleri daha sınırlı bir alanı temsil etmektedir. Bu durum genel olarak konu modellemede karşılaşılan bir durumdur. Yazarların çeşitli sınıflandırma şemalarına göre (örneğin SWEBOK konu alanları gibi) yaptıkları sınıflandırmalarda konuların detay seviyelerinin birbirine daha yakın olması beklenebilir. Ancak bu yöntemlerle de çok sayıda yayını sınıflandırmanın maliyeti çok yüksektir.

6 Sonuçlar

Bu çalışma kapsamında 1989 ile 2017 arasında YM alanında yayımlanmış 32.701 makale ve bildiriden oluşan bir belge kümesi kullanılarak YM alanına özgü 11 konudan oluşan bir konu modeli oluşturulmuştur. Bu konu modeli kullanılarak UYMS'de 2007 ile 2018 arasında sunulan 564 bildiri analiz edilmiştir. Elde edilen sonuçlar doğrultusunda YM alanında Türkiye'deki araştırmacıların üzerinde çalıştığı konular yıllar boyunca belirlenmiştir. Bu bulgular literatürdeki diğer benzer çalışmalardaki bulgularla karşılaştırılarak yorumlanmıştır.

Kaynakça

1. Garousi, V. (2015). A bibliometric analysis of the Turkish software engineering research community. Scientometrics, 105(1), 23-49.
2. Blei, D.M. (2012). Probabilistic topic models. Commun. ACM 55, 4 (April 2012), 77-84. doi: https://doi.org/10.1145/2133806.2133826
3. Hofmann, T. (1999, July). Probabilistic latent semantic analysis. In Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence (pp. 289-296). Morgan Kaufmann Publishers Inc..
4. Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. Journal of machine Learning research, 3(Jan), 993-1022.
5. Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. Proceedings of the National academy of Sciences, 101(suppl 1), 5228-5235.
6. Bittermann, A., & Fischer, A. (2018). How to identify hot topics in psychology using topic modeling. Zeitschrift für Psychologie.
7. Kim, Y. M., & Delen, D. (2018). Medical informatics research trend analysis: A text mining approach. Health informatics journal, 24(4), 432-452.
8. Hall, D., Jurafsky, D., & Manning, C. D. (2008, October). Studying the history of ideas using topic models. In Proceedings of the conference on empirical methods in natural language processing (pp. 363-371). Association for Computational Linguistics.
9. Glass, R. L., Vessey, I., & Ramesh, V. (2002). Research in software engineering: an analysis of the literature. Information and Software technology, 44(8), 491-506.
10. Cai, K. Y., & Card, D. (2008). An analysis of research topics in software engineering–2006. Journal of Systems and Software, 81(6), 1051-1058.
11. Demir, K.A., Mut, A., Okçu, H., Alpyavuz, O. (2011). Türkiye'deki Yazılım Mühendisliği Konferansları Üzerine Bir İnceleme. 5. Ulusal Yazılım Mühendisliği Sempozyumu.
12. Wang, Z., Li, B., & Ma, Y. (2014). An Analysis of Research in Software Engineering: Assessment and Trends. arXiv preprint arXiv:1407.4903.
13. Raulamo-Jurvanen, P., Mantyla, M. V., & Garousi, V. (2015, October). Citation and Topic Analysis of the ESEM papers. In 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (pp. 1-4). IEEE.
14. Turdaliev, N., Bilgin, B., Deniz, G., Durdu, P. O., Incebacak, D., & Mutlu, A. (2015). UYMS Araştırma Eğilimleri: Bir Sistematik Eşleme Çalışması. 9. Ulusal Yazılım Mühendisliği Sempozyumu.
15. Garousi, V., & Mäntylä, M. V. (2016). Citations, research topics and active countries in software engineering: A bibliometrics study. Computer Science Review, 19, 56-77.
16. Mathew, G., Agrawal, A., & Menzies, T. (2018). Finding Trends in Software Research. IEEE Transactions on Software Engineering. doi: 10.1109/TSE.2018.2870388
17. Rehurek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. In In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks.
18. Röder, M., Both, A., & Hinneburg, A. (2015, February). Exploring the space of topic coherence measures. In Proceedings of the eighth ACM international conference on Web search and data mining (pp. 399-408). ACM.

Web Erişilebilirlik Değerlendirmesi: Sistematik Haritalama

Ömer Naci Soydemir[1,*], Pınar Onay Durdu[2]

[1] Bilgisayar Mühendisliği Bölümü, Fen Bilimleri Enstitüsü Kocaeli Üniversitesi, Kocaeli, Türkiye
soydemir.naci@gmail.com
[2] Bilgisayar Mühendisliği Bölümü, Mühendislik Fakültesi Kocaeli Üniversitesi, Kocaeli, Türkiye
pinar.onaydurdu@kocaeli.edu.tr

Özet. Erişilebilirlik, herhangi bir ürünün, servisin, hizmetin, teknolojinin ya da uygulamanın engelliler ve yaşlılar dahil olmak üzere herkes tarafından anlaşılır ve kullanılabilir olmasını ifade etmektedir. Web uygulamalarında ise bu kavram, hedef kullanıcı kitlesinin tamamının uygulamaya erişebilmesi, uygulamayı kullanabilmesi ve içeriği anlayabilmesini ifade etmektedir. Geliştiricilerin belli bir standart üzerinden erişilebilir bir uygulama ortaya çıkarmaları için, uluslararası web erişilebilirlik standardı olan teknik ve yönergeler kapsamında uygulanacak kontroller tanımlanmıştır. Bu çalışmada 2008-2018 yılları arasında yapılmış web erişilebilirlik değerlendirmesi ve ölçütleri ile alakalı 110 çalışma sistematik haritalama yöntemi ile incelenmiştir. Çalışmalar incelendiğinde, yayınlanan çalışmaların günümüze geldikçe sayılarının arttığı, en fazla web erişilebilirlik değerlendirme yöntemi olarak otomatik değerlendirme araçlarının kullanıldığı, sıklıkla e-devlet uygulamalarına yönelik değerlendirme çalışmalarının yapıldığı ve web erişilebilirlik ölçütü olarak WAB'ın kullanıldığı gözlemlenmektedir.

Anahtar Kelimeler: Web Erişilebilirliği, Web Erişilebilirlik Değerlendirmesi, Web Erişilebilirlik Ölçütü, Sistematik Haritalama.

Web Accessibility Evaluation: Systematic Mapping

Ömer Naci Soydemir[1,*], Pınar Onay Durdu[2]

[1] Computer Engineering Department, Institute of Science and Technology, Kocaeli University, Kocaeli, Turkey
soydemir.naci@gmail.com
[2] Computer Engineering Department, Engineering Faculty, Kocaeli University, Kocaeli, Turkey
pinar.onaydurdu@kocaeli.edu.tr

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Abstract. Accessibility means that any product, service, technology or application can be understood and used by everyone, including the disabled and the elderly. In web applications, this concept means that the entire target audience can access the application, use the application, and understand the content. There is international web accessibility standard that guide the developers by defining the necessary controls to develop accessible applications. In this study, 110 papers related to web accessibility evaluation and metrics published between 2008 and 2018 were examined by systematic mapping and the findings were reported. The results revealed that the number of published papers were increased since 2008, automatic accessibility evaluation methods were the most common method applied in the studies, generally e-government web sites were evaluated and WAB is also used as web accessibility metric.

Keywords: Web Accessibility, Web Accessibility Evaluation, Web Accessibility Metric, Systematic Mapping.

1 Giriş

Web siteleri zaman ve mekândan bağımsız erişim sağlama özellikleri ile engelli kişilere bilgi ulaşımı ve iletişimi açısından eşi görülmemiş bir olanak sağlamaktadır [1]. Bununla beraber web sitelerinin tüm kullanıcılar için erişilebilir olması sağladıkları hizmetlerden etkin şekilde faydalanabilmek için gerekli ön koşullardan biridir. Bunun yanı sıra kuruluşlar için, yasal riskleri azaltmak, marka varlığını güçlendirmek, müşteri deneyimini ve iş verimliliğini artırmak gibi birçok avantajı da beraberinde getirmektedir [2].

Dünya sağlık örgütünün rakamlarına göre dünya nüfusunun yaklaşık %15'inin engelli olduğu tahmin edilmektedir [3]. Web sitelerinin vazgeçilmez hale gelmesi nedeniyle de web ortamını daha erişebilir kılmak için pek çok ülkede Web içeriği erişilebilirliği yönergesi (WCAG (Web Content Accessibility Guidelines)) 1.0 [4] veya 2.0 [5] gibi genel ilkelere ek olarak, ISO 9241-20 [6] veya Section 508 [7] gibi yasa ve yönetmelikler uygulanmaktadır. WCAG 2.0, W3C (World Wide Web Consortium) tarafından hazırlanmıştır ve web sayfalarının geliştirilmesi sırasında, tüm kullanıcılar için belirli bir erişilebilirlik düzeyi sağlamak için geliştiriciler, tasarımcılar vb. tarafından takip edilebilecek 91 adet doğrulanabilir kontrol kriteri içermektedir. Bu kriterler için A, AA ve AAA olarak adlandırılan üç öncelik seviyesi tanımlanmıştır. Tablo 1'de özetlendiği şekilde öncelik 1 ya da A düzeyi olarak tanımlanan düzey web sitelerinin uyması gereken minimum uyumluluk düzeyini belirtmektedir ve gerçekleşmesi gereken 29 kontrol maddesi içermektedir. AA düzeyinde (Öncelik 2) 40 kontrol; AAA düzeyinde (Öncelik 3) ise 22 adet kontrol yer almaktadır.

Tablo 1. WCAG 2.0 (2008) Öncelik Seviyelerinin Kapsamı

Öncelik Seviyesi	Kriter Kapsamı	Kriter Sayısı
A Düzeyi	Temel seviyede yerine getirilmesi beklenen standartlar	29

AA Düzeyi	Geliştiricilere, yerine getirilmesi tavsiye edilen standartlar	40
AAA Düzeyi	Yerine getirilmesi ideal olan standartlar	22

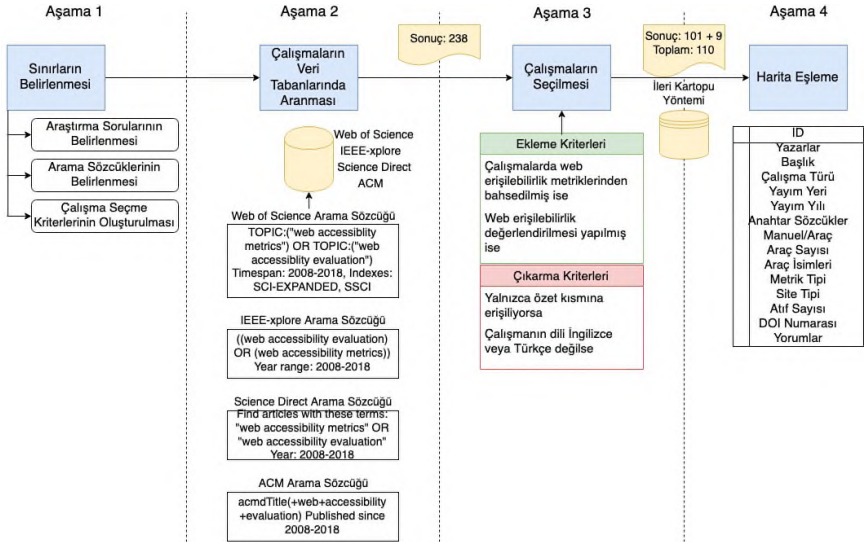
Web sitelerinin erişilebilirlik değerlendirmeleri sıklıkla web sitesinin WCAG yönergelerinde tanımlanan kontrol kriterlerine ne kadar uyumlu olduğunun çeşitli otomatik araçlar [8] ile ya da manuel olarak [9] ya da her iki yöntemin de beraber kullanılarak [10] belirlenmesi ile gerçekleştirilmektedir. Yapılan değerlendirmelerin sonucunda web sitesinin nicel olarak erişilebilirlik seviyesini göstermek için ise çeşitli erişilebilirlik metrikleri önerilmiştir [11]. Metrikler, yazılım geliştirmede ürün ve süreçlerin anlaşılmasına, bu süreçlerin kontrolüne ve iyileştirilmesine yardımcı olmak için önemlidir [12]. Erişilebilirlik açısından ise metrikler mevcut ürünlerin erişilebilirliğinin artırılması ve geliştirme sürecindeki bir üründeki erişilebilirlik sorunlarının tanımlanması konularında faydalıdır. Web erişilebilirliğinin ölçülmesinde metrik tanımlama ve bu metrikleri uygulama konusunda da çeşitli çalışmalar gerçekleştirilmiştir [13].

Alan yazında gerçekleştirilen pek çok erişilebilirlik değerlendirme ve erişilebilirlik metrik önerisi içeren çalışma bu alanda çalışan geliştirici ve araştırmacılar için oldukça değerlidir. Ancak bu çalışma kapsamında özellikle web erişilebilirlik değerlendirmeleri ve metrikleri üzerine yapılmış olan bu birincil çalışmalar bir araya getirilerek ikincil bir çalışma yöntemi olan sistematik haritalama yöntemi ile araştırma alanıyla alakalı olarak güncel eğilimlerin belirlenmesi hedeflenmiştir. Sistematik haritalama yöntemi, bir araştırma konusunu sistematik olarak analiz etmeyi ve bu alanla ilgili araştırmacılara genel bir fikir vermeyi hedefleyen bir yaklaşımdır [14]. Bu amaçla gerçekleştirilen önceki çalışmalar incelendiğinde çalışmalar genellikle ya belli uygulama alanlarındaki genel erişilebilirlik çalışmalarına ya da sadece web erişilebilirlik metriklerine odaklandıkları görülmektedir. Örneğin [15] büyük çevrim içi kurs (MOOC) platformlarındaki ve [16] Suudi Arabistan'daki üniversite ve devlet web sitelerindeki genel erişilebilirlik konularına odaklanan sistematik alan yazın incelemeleridir. Ayrıca [17]'de özellikle bilişsel engeli bulunan kullanıcılara yönelik elektronik iletişim erişilebilirliğine yönelik ölçümlere odaklanılan bir çalışmanın bulguları raporlanmaktadır. Diğer taraftan [11] web erişilebilirlik ölçütlerini niteliklerine göre değerlendirdikleri çalışmalarında o zamana kadar önerilen çeşitli web erişilebilirlik metrikleri derlemişlerdir. [18] ise yine web erişilebilirlik metriklerini kıyaslamak için bir çerçeve önerdikleri çalışmalarında yine kapsamlı bir metrik listesi sunmaktadır. Bu çalışmalar incelendiğinde web erişilebilirlik değerlendirmeleri ve metriklerini bir arada değerlendirmeye yönelik gerçekleştirilmiş bir çalışma bulunmamaktadır. Bu doğrultuda bu çalışma kapsamında her iki konuyu da kapsayacak şekilde bir değerlendirmenin gerçekleştirilmesi hedeflenmiştir.

2 Yöntem

Bu çalışma kapsamında kanıta dayalı yazılım mühendisliğine [19] dayalı olarak geliştirilmiş olan sistematik haritalama yöntemi [14] uygulanmaktadır. Bu yöntem belirli bir araştırma konusunda genel bir fikir verme ve o alanda yapılmış çalışmaların miktarı, tipi ve sonuçları ile ilgili bilgi verme şeklinde özetlenmektedir.

Haritalama çalışmasını uygulayabilmek için, Kitchenham vd. [19]'nin çalışmalarında önerdiği gözden geçirme, inceleme ve raporlama aşamalarına benzer şekilde 4 adımlı bir süreç takip edilmiştir. Uygulanan araştırma süreci Şekil 1'de görülmektedir. İlk aşama olan sınırların belirlenmesinde çalışmayı tanımlayan çerçeveler oluşturulmuştur. Takip eden aşamalarda yapılan birincil çalışmaları tanımlamak, anlamlı veriler çıkarmak ve sınıflandırmak için gerekli adımlar gerçekleştirilmiştir. Bu çalışma WCAG 2.0'ın [5] yayınlanma yılı olan 2008 yılından 2018'e kadar olan çalışmaları kapsayacak şekilde gerçekleştirilmiştir.



Şekil 1. Sistematik Haritalama Aşamaları

2.1 Sınırların Belirlenmesi

Çalışmanın sınırlarının belirlenmesi aşamasında aşağıdaki adımlar uygulanmıştır;

- Araştırma sorularının tanımlanması: Araştırmanın amacı doğrultusunda web erişilebilirlik değerlendirmesi ve web erişilebilirlik metriklerine yönelik güncel durumu ve eğilimleri belirlemeye yönelik olarak Tablo 2'de listelenen araştırma soruları tanımlanmıştır.

Tablo 2. Araştırma Soruları

Sayı	Açıklama
1	Yayınların yıllara göre dağılımı, yayın türleri ve yayınlanma yerleri nasıl değişmektedir?
2	Kullanılan erişilebilirlik değerlendirme yöntemleri nelerdir?
3	Kullanılan otomatik değerlendirme araç sayıları nasıl değişmiştir?
4	Sık kullanılan otomatik değerlendirme araçları hangileridir?
5	Değerlendirme yapılan web site türleri nasıl değişmektedir?

6	Çalışmalarda en sık hangi ölçütler kullanılmıştır?

- Arama sözcüklerinin belirlenmesi: Çalışmanın amacı doğrultusunda birincil çalışmaları bulmak için aramalarda kullanılacak genel anahtar sözcükler; "web erişilebilirlik değerlendirmesi (web accessibility evaluation)" ve "web erişilebilirlik metrikleri (web accessibility metrics)" belirlenmiştir. Ayrıca birincil kaynakların seçimi için kullanılabilirlik, erişilebilirlik ve kalite kriterlerini dikkate almak önemlidir [10].
- Ekleme ve çıkarma kriterlerinin tanımlanması: Uygulanmasının kolay olması düşünülerek Tablo 3'de tanımlanmış olan birincil çalışmaları ekleme ve çıkarma kriterleri benimsenmiştir. Burada belirtilen ekleme kriterlerinden birinin sağlanması durumunda çalışma incelenmek üzere eklenilmiştir.

Tablo 3. Ekleme ve Çıkarma Kriterleri

Tür	**Açıklama**
Ekleme	Çalışmada web erişilebilirlik metriklerinden bahsedilmiş ise (uygulanmış ise)
	Web erişilebilirlik değerlendirilmesi yapılmış ise
Çıkarma	Çalışmanın yazım dili İngilizce veya Türkçe değilse
	Yalnızca özet kısmına erişiliyorsa

2.2 Çalışmaların Veri Tabanlarında Aranması

Bu aşamada başlangıçta Tablo 4'de belirlenmiş bilimsel veri tabanlarından ilk dördünde arama işlemi gerçekleştirilmiştir. Bu adımı gerçekleştirebilmek için belirtilen bilimsel veri tabanları tarafından sağlanan arama motorları kullanılmıştır. Tablo 4'de veri tabanlarında çalışma amacı ile alakalı arama sözcükleri, sonuç olarak dönen makale sayısı ve 2.3 bölümün-de detaylı bahsedilecek olan kriterlerden sonra kalan makale sayısı belirtilmektedir. Bölüm 2.4 de açıklanan ileri kartopu yöntemi kapsamında başta kullanılan veri tabanlarına Google Scholar da eklenmiştir.

Tablo 4. Veri Tabanları Arama Sözcükleri ve Sonuç Sayısı

	Dönen Makale Sayısı	**İncelenen Makale Sayısı**
Web of Science	61	30
Science Direct	41	23
IEEE-xplore	74	22
ACM	62	26
Google Scholar	70	9
Toplam	308	110

2.3 Çalışmaların Seçilmesi

Bu aşamada araştırma sorularına cevap verebilme adına incelenecek birincil çalışmaları belirlemek için Tablo 3'de açıklanan seçim kriterleri uygulanmıştır. Bu

nedenle 2.2 bölümünde toplanan tüm çalışmalar başlık, özet ve anahtar kelimeleri incelenmiştir. Bu kısımlardan anlaşılmama gerçekleşirse çalışmanın tümü okunmuştur. Bu kriterler uygulandıktan sonra kalan çalışma sayıları Tablo 4'de paylaşılmıştır.

2.4 İleri Kartopu Yönteminin Gerçekleştirilmesi

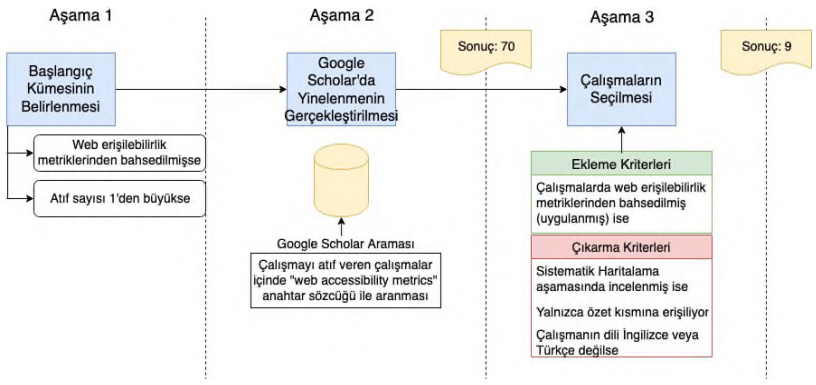
İncelenecek araştırmaların mümkün olduğunca gözden kaçmaması için sistematik haritalama aşamaları ileri kartopu yöntemi de uygulanarak desteklenmiştir. Bu yöntem, bir çalışmaya ait referans ve alıntıları kullanarak daha fazla ilgili çalışmaya ulaşılabilmeyi sağlamaktadır. Bununla birlikte kartopu sadece referans listeleri ve alıntılara bakmakla kalmaz, aynı zamanda çalışmaları referans verilen yerlere ve makalenin nereye atıf yapıldığına bakmanın sistematik bir yolu ile tamamlayabilir [15].

Sistematik Haritalama aşamasında incelenen ve şu koşulları sağlayan çalışmalar ile ileri kartopu aşaması için başlangıç kümesi oluşturulmuştur;

- Web erişilebilirlik metrik uygulanmasını içeren çalışmalar,
- Atıf sayısı 1'den fazla olan çalışmalar.

Başlangıç kümesine dahil edilen çalışmalara atıf veren çalışmalar Google Scholar ile tanımlanmıştır. İlgili alt çalışmalar üzerinde ise yine Google Scholar üzerinden "web accessibility metrics" anahtar sözcüğü ile arama gerçekleştirilmiştir. Çalışmanın dahil edilip edilmemesine ise;

- Özet kısmının incelenmesi,
- Özet yeterli bilgi içermiyorsa atıf verilen yer ile beraber diğer kısımların incelenmesi, gerçekleştirilerek karar verilmiştir.



Şekil 2. İleri Kartopu Yöntemi Aşamaları

2.5 Analiz ve Haritalama

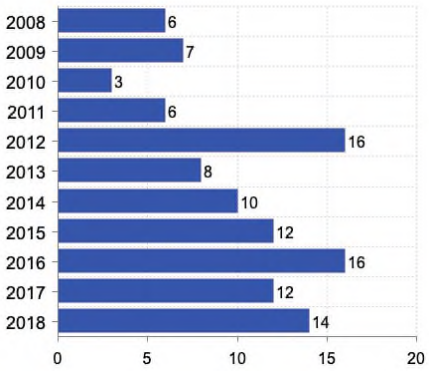
Eşleme formu tanımlanan araştırma sorularına ve seçilen kaynak çalışmaların bu soruları cevaplamak için nasıl kullanılacağını planlama amacıyla hazırlanmıştır. Haritalamada kullanılan eşleme formları ve bu çalışmada incelenen dokümanların listesi Google Drive üzerinde yer almaktadır *.

3 Bulgular

Bu çalışma kapsamında, web erişilebilirlik değerlendirmesi ve metrikleri ile alakalı bilimsel veri tabanlarında yayınlanan çalışmalara ait bir sistematik eşleme çalışması gerçekleştirilmiştir. Sınıflandırma amacıyla Bölüm 3.1'de açıklanan detaylı inceleme protokolü kullanılarak yaklaşık 10 yıllık bir dönemde gerçekleştirilen araştırma çalışmalarına ait eğilimler belirlenmeye çalışılmıştır.

3.1 Yayınların yıllara göre dağılımı, yayın türleri ve yayınlanma yeri

İncelenen çalışmaların yayınlanma yıllarına göre dağılımı Şekil 3'de gösterilmiştir. WCAG 2.0 standartlarının yayınlanma tarihi olan 2008 [5] yılından 2018 yılına kadar incelenen çalışmalarda 2012 yılında yoğunluğun arttığı ve 2012 yılından sonrasında önceki çalışmalara oranla daha fazla çalışma gerçekleştiği gözlemlenmiştir. 2016 yılında ise yapılan çalışmalar en yüksek noktasına ulaşmıştır.



Şekil 3. Yıllara Göre Yayınlanan Çalışmaların Dağılımı

İncelenen çalışmaların eser türleri Tablo 6'da belirtilmiştir. Web erişilebilirlik değerlendirmesi ile yapılan yayınların daha çok bilimsel kongrelerde yapıldığı ancak son yıllara doğru bu konuyla ilgili bilimsel dergilerde de yayın sayısının arttığı gözlenmektedir.

Tablo 5. Çalışmaların Eser Türleri

Yıllar	Bilimsel Dergi Makalesi	Konferans Bildirisi
2008	1	5
2009	2	5
2010	-	3
2011	2	4
2012	4	12
2013	3	5
2014	4	6
2015	6	6

2016	8	8
2017	4	8
2018	3	11

Sıklıkla yayın yapılan dergiler incelendiğinde Tablo 6'de belirtildiği gibi kapsamında insan-bilgisayar etkileşimi konusu olan bilimsel dergiler gözlemlenmiştir. Konferans bildirilerinin konu başlıkları olarak insan-bilgisayar etkileşiminin yanı sıra yazılım mühendisliği ve çok disiplinli yazılım teknolojileri konularının da olduğu Tablo 7'den görülebilmektedir.

Tablo 6. Birden Fazla Makale Yayımlanan Dergiler

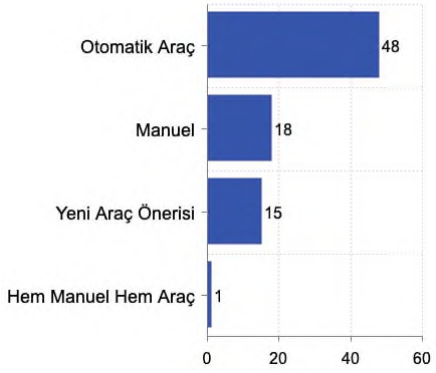
Bilimsel Dergi Adı	**Makale Sayısı**
Universal Access in the Information Society	4
Universal Access in Human-Computer Interaction	3
Computers in Human Behavior	3
Government Information Quarterly	2

Tablo 7. Birden Fazla Yayınların Sunulduğu Konferanslar

Konferans Adı	**Makale Sayısı**
International Cross-Disciplinary Conference on Web Accessibility	12
International Conference on Software Development for Enhancing Accessibility and Fighting Info-exclusion	5
Procedia Computer Science	5
Web for All Conference on The Future of Accessible Work	4
Web for All Conference	3
International Conference on eDemocracy & eGovernment (ICEDEG)	3
Proceedings of the Internet of Accessible Things	3
International Conference on Information and Communication Technology and Accessibility	2
International Conference on User Science and Engineering (i-USEr)	2
International Conference on Computer Science and Software Engineering	2

3.2 Kullanılan erişilebilirlik değerlendirme yöntemleri

İncelenen çalışmalarda web erişilebilirlik değerlendirilmelerinin gerçekleştirme yöntemi Şekil 4'de görülmektedir. Değerlendirmelerde otomatik araç kullanımı en sık tercih edilen yöntem olmuştur. Yeni araç önerilerinde mevcut otomatik araçlar ile diğer araçların karşılaştırmaları da gerçekleştirilmiştir. Bu karşılaştırmalar 3.3 bölümündeki otomatik araç sayılarına eklenmiştir.



Şekil 4. Değerlendirmelerde Tercih Edilen Yöntemler

3.3 Kullanılan otomatik değerlendirme araç sayıları

Tablo 8'dan görüldüğü üzere otomatik araç kullanımlarında çoğu web erişilebilirlik değerlendirilmesi tek araç kullanımı ile gerçekleştirilmiştir. Fakat alan yazında önerildiği gibi [22] en az 2 veya daha fazla araç ile değerlendirme toplam sayısı da tek araç ile değerlendirme sayısına yakındır.

Tablo 8. Değerlendirmelerde Kullanılan Otomatik Araç Sayısı

Araç Sayısı	1	2	3	4	5	6	7
Kullanım Sayısı	27	13	4	2	2	1	1

3.4 Sık kullanılan otomatik değerlendirme araçları

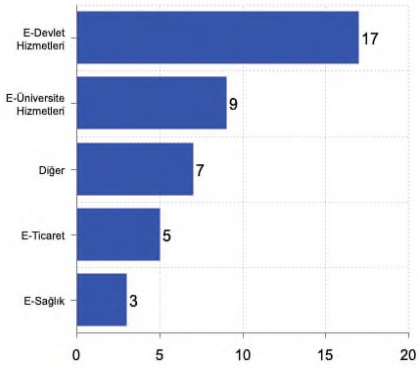
Web erişilebilirlik değerlendirmelerinde kullanılan otomatik araç türleri çeşitlilik göstermektedir. Tablo 9'da araştırmalarda 3 veya daha sık kullanıldığı tespit edilen araçlar görülmektedir ve gözlemlendiği gibi Wave ve AChecker araçları en çok tercih edilenlerdir. Burada bu araçların ara yüzlerinin araştırmacılara detaylı ve anlamlı sonuçlar sunmasının etkisi büyüktür.

Tablo 9. Değerlendirmelerde Kullanılan Otomatik Araçlar

Araç Adı	**Kullanım Sayısı**	**Araç Adı**	**Kullanım Sayısı**
Wave	18	Eval Access	4
Achecker	16	EIII Page Checker	3
TAW	9	Examinator	3
Total Validator	6	Qual Web	3
HERA	5	Sort Site	3

3.5 Değerlendirme yapılan web site türleri

İncelenen çalışmalarda web erişilebilirlik değerlendirmesi yapılan site türlerinde e-Devlet hizmetleri sunan web siteleri Şekil 5'de gözlemlendiği gibi çalışmaların büyük bir kısmını kapsamaktadır. Burada üniversite web site üzerine yapılan çalışmaların sayısı da dikkat çekmektedir. Diğer kategorisinde ise genel olarak dağınık web site türleri yer almakta olup, restoran, otel, kurumsal web siteleri gibi çeşitlilik göstermektedir. Bu değerlendirmeleri gerçekleştiren makalelerin çalışma numaraları Google Drive üzerinde yer almaktadır. *



Şekil 5. Değerlendirilen Web Site Türleri

3.6 Çalışmalarda sık kullanılan metrikler

İncelenen çalışmalarda kullanılan web erişilebilirlik ölçütlerinin en sık kullanılanları Tablo 10'de sunulmuştur. En sık kullanılan metriklerin WAB (Web Accessibility Metric), WAQM (Web Acessiibility Quantitative Metric) ve UWEM (The Unified Web Evaluation Methodology) dir. Çok sayıda bireysel çalışmada kullanılan ya da önerilen ölçüt de bulunmaktadır ve bunlar diğer kategorisi ile gruplanmıştır. Bu metrikleri kullanan makalelerin çalışma numaraları Google Drive üzerinde yer almaktadır. *

Tablo 10. Çalışmalarda Karşılaşılan Web Erişilebilirlik Metrikleri

Metrik İsmi	**Kullanım Sayısı**
WAB	6
WAQM	4
UWEM	4
WCAG 2.0 Metrikleri	2
WABS	2
GQM	2
F-Measure	2
Diğer Metrikler	16

4 Sonuç ve Değerlendirme

Bu çalışmada web erişilebilirlik değerlendirmeleri ve metrikleri ile ilgili olan 2008-2018 yılları arasında yayınlanan 110 adet çalışma sistematik haritalama yöntemi ile incelenmiştir. Araştırma sonuçlarına göre günümüze yaklaştıkça yayımlanan çalışmaların arttığı ve en çok yayının da bilimsel konferanslarda yapıldığı görülmektedir. Çalışmaların eser türleri ile beraber yayınlandıkları dergi ve konferanslar bu konu ile araştırma yapmak isteyen araştırmacıların faydalanması amacıyla belirtilmiştir. Web erişilebilirlik değerlendirme yöntemi olarak çalışmalarda çoğunlukla otomatik değerlendirme aracı kullanıldığı, bu araçlardan da Wave ve AChecker araçlarının en çok tercih edilen araçlar olduğu göze çarpmaktadır. Değerlendirme yapılan site türlerinden e-devlet hizmetleri en sık değerlendirilen web site türü olarak karşımıza çıkmaktadır. Web erişilebilirlik ölçütleri konusunda ise en sık tercih edilen metriklerin, WAB, UWEM ve WAQM olduğu gözlemlenmiştir. Bu çalışmanın sonuçlarının web erişilebilirlik değerlendirmesi ve web erişilebilirlik ölçütleri konusunda güncel eğilimlerin ortaya çıkarılmasında, hangi konular üzerinde daha çok çalışıldığının, çoğunlukla hangi yöntemlerin kullanıldığının belirlenmesi ile bu konuda çalışan geliştirici ve araştırmacılara yol gösterici olacağına inanılmaktadır. Çalışma kapsamında tespit edilen web erişilebilirlik ölçütlerinin detaylı bir incelemeye tabi tutulması da bundan sonra yapılacaklar arasında yer almaktadır.

Bu tip çalışmaların en önemli kısıtlarından biri güvenilirliktir. Bu amaçla çalışma tekrarlı bir süreç ile gerçekleştirilmiştir. Araştırmacılar çalışmaları önce tek başlarına incelemişler daha sonra bulgularını bir araya getirerek ortak fikir birliğine varmışlardır. Böylece tek bir inceleyiciye dayalı olabilecek yanlılık engellenmeye çalışılmıştır. Diğer bir kısıt da çalışmaya tüm yayınlarının dahil edilememiş olma olasılığı ve özellikle ölçütler ile ilgili çalışmaların gözden kaçma olasılığıdır. Ancak bunun önüne geçmek amacıyla bu konuda yayınların indekslendiği bilimsel veri tabalarının pek çoğu çalışmaya dahil edilmiş ve son olarak da ileri kartopu yöntemi ile Google Scholar'dan alakalı çalışmaların dahil edilmesi sağlanmıştır.

* https://drive.google.com/open?id=14bekCZeC4VueVxjs0--or3kWV5Gd6ZaG

Kaynakça

1. Web Accessibility Initiative (WAI). (2019). Introduction to Web Accessibility. [online] Available at: https://www.w3.org/WAI/fundamentals/accessibility-intro/#important [Accessed 29 May 2019].
2. Web Accessibility Initiative (WAI). (2019). The Business Case for Digital Accessibility. [online] Available at: https://www.w3.org/WAI/business-case/ [Accessed 29 May 2019].
3. World Health Organization. (2019). World report on disability. [online] Available at: https://www.who.int/disabilities/world_report/2011/report/en/ [Accessed 29 May 2019].
4. W3.org. (2019). Web Content Accessibility Guidelines 1.0. [online] Available at: https://www.w3.org/TR/WAI-WEBCONTENT/ [Accessed 29 May 2019].
5. W3.org. (2019). Web Content Accessibility Guidelines (WCAG) 2.0. [online] Available at: https://www.w3.org/TR/WCAG20/ [Accessed 29 May 2019].

6. Iso.org. (2019). [online] Available at: https://www.iso.org/obp/ui/#iso:std:iso:9241:-20:ed-1:v1:en [Accessed 29 May 2019].
7. Section508.gov. (2019). Section508.gov | GSA Government-wide IT Accessibility Program. [online] Available at: https://www.section508.gov/ [Accessed 29 May 2019].
8. Zaphiris, P., Ellis, R.D.: Website Usability and Content Accessibility of the top USA Universities. 6.
9. Alahmadi, T., Drew, S.: Accessibility Evaluation of Top-ranking University Websites in World, Oceania, and Arab Categories for Home, Admission, and Course Description Webpages. 18.
10. ITD Journal: Research On Web Accessibility In Higher Education. [online] Available at: http://itd.athenpro.org/volume9/number2/thompson.html [Accessed 29 May 2019].
11. Freire, A.P., Fortes, R.P.M., Turine, M.A.S., Paiva, D.M.B.: An evaluation of web accessibility metrics based on their attributes. In: Proceedings of the 26th annual ACM international conference on Design of communication - SIGDOC '08. p. 73. ACM Press, Lisbon, Portugal (2008). https://doi.org/10.1145/1456536.1456551.
12. N Fenton, N., Bieman, J.: Software Metrics. 77.
13. Sullivan, T., Matson, R.: Barriers to use: usability and content accessibility on the Web's most popular sites. In: Proceedings on the 2000 conference on Universal Usability - CUU '00. pp. 139–144. ACM Press, Arlington, Virginia, United States (2000). https://doi.org/10.1145/355460.355549.
14. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic Mapping Studies in Software Engineering. 10.
15. Sanchez-Gordon, S., Luján-Mora, S.: Research challenges in accessible MOOCs: a systematic literature review 2008–2016. Univers. Access Inf. Soc. 17, 775–789 (2018). https://doi.org/10.1007/s10209-017-0531-2.
16. Akram, M., Bt, R.: A Systematic Literature Review to Determine the Web Accessibility Issues in Saudi Arabian University and Government Websites for Disable People. Int. J. Adv. Comput. Sci. Appl. 8, (2017). https://doi.org/10.14569/IJACSA.2017.080642.
17. Borg, J., Lantz, A., Gulliksen, J.: Accessibility to electronic communication for people with cognitive disabilities: a systematic search and review of empirical evidence. Univers. Access Inf. Soc. 14, 547–562 (2015). https://doi.org/10.1007/s10209-014-0351-6.
18. Vigo, M., Arrue, M., Brajnik, G., Lomuscio, R., Abascal, J.: Quantitative metrics for measuring web accessibility. In: Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A) - W4A '07. p. 99. ACM Press, Banff, Canada (2007). https://doi.org/10.1145/1243441.1243465.
19. Kitchenham, B.A., Dyba, T., Jorgensen, M.: Evidence-based software engineering. In: Proceedings. 26th International Conference on Software Engineering. pp. 273–281. IEEE Comput. Soc, Edinburgh, UK (2004). https://doi.org/10.1109/ICSE.2004.1317449.
20. Budgen, D., Turner, M., Brereton, P., Kitchenham, B.: Using Mapping Studies in Software Engineering. 10.
21. Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14.
22. Web Accessibility Initiative (WAI). (2019). Selecting Web Accessibility Evaluation Tools. [online] Available at: https://www.w3.org/WAI/test-evaluate/tools/selecting/ [Accessed 29 May 2019].

Çalışma Sırasında Konfigüre Edilebilir Ortak Gömülü Simülatör

Erkan TAN[1], İbrahim BALCI[1], Uğur YILMAZ[1], Elifnur ÖZTÜRK[1], Derya Yeliz ULUTAŞ[1]

[1]ASELSAN A.Ş. Ankara, Türkiye

{erkantan, ibalci, uguryilmaz, elifozturk, dyulutas}@aselsan.com.tr

Özet. Gömülü sistemlerde farklı arayüzlerin kullanımı yazılım testlerinde kullanılan simülatör çeşitliliğini ve sayısını arttırmaktadır. Artan simülatör sayısı ile kara kutu testler sırasında gerçekleştirilen senaryosal testlerde kullanılan simülatörlerin kontrolü zorlaşmaktadır. Ayrıca her bir arayüz için ayrı bir simülatör geliştirilmesi testler için harcanan iş gücünü arttırmaktadır. Bu sebeplerden yola çıkılarak simülatörlerin kontrolünün kolaylaştırılması ve ortak bir çerçevede toplanması amacıyla Ortak Gömülü Simülatör (OGS) geliştirilmiştir. Bu simülatör sayesinde farklı arayüzler (MsgQ, PCIe, TCP, UDP, seri kanal) tek bir noktadan yönetilebilir ve gözlemlenebilir hale getirilmiştir.

Anahtar Kelimeler*:* Simülatör, Yazılım test, Gömülü Yazılım, Yeniden Kullanılabilirlik

Run-Time Configurable Common Embedded Simulator

Erkan TAN[1], İbrahim BALCI[1], Uğur YILMAZ[1], Elifnur ÖZTÜRK[1], Derya Yeliz ULUTAŞ[1]

[1]ASELSAN A.Ş. Ankara, Turkey

{erkantan, ibalci, uguryilmaz, elifozturk, dyulutas}@aselsan.com.tr

Abstract. The use of different interfaces in embedded systems increases the number and variety of simulators used in software testing. With the increasing number of simulators, it is difficult to control the simulators used in the scenario tests performed during the black box tests. In addition, the development of a separate simulator for each interface increases the workforce spent for testing. Based on these reasons, Common Embedded Simulator (OGS) has been developed in order to facilitate the control of the simulators and to gather them in a common framework. With this simulator, different interfaces (MsgQ, PCIe, TCP, UDP, serial channel) can be managed and monitored from a single point.

Keywords: Simulator, Software Testing, Embedded Software, Reusability

1 Giriş

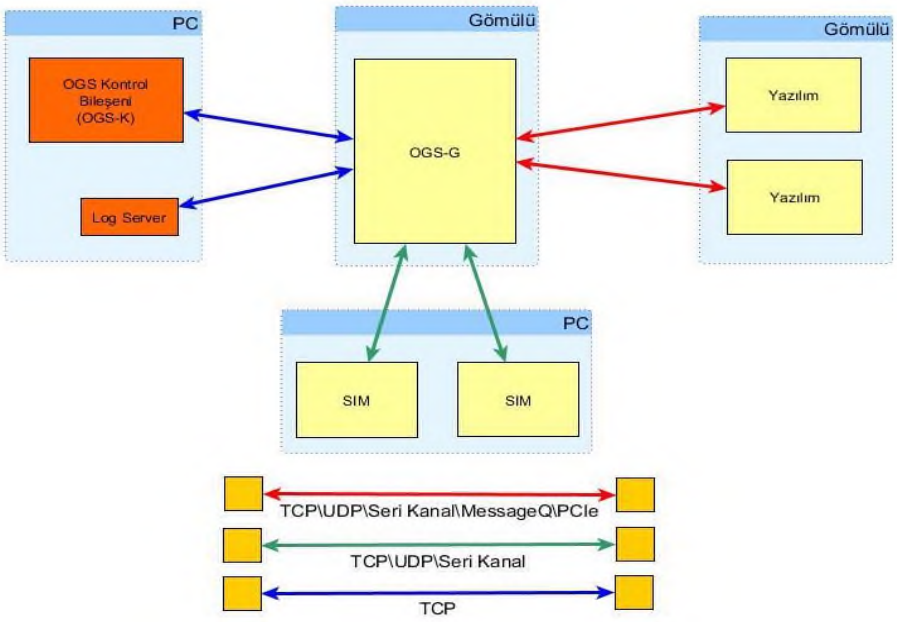
Aselsan Radar ve Elektronik Harp Sistemleri Sektör Başkanlığı(REHIS) Yazılım Test Mühendisliği Müdürlüğü, yazılım test yöntemi olarak "Black Box" (kara kutu) test metodunu kullanmaktadır. Kara kutu testler gereksinim ve fonksiyonellik üzerinedir. Test edilecek yazılımın kaynak kodu ya da tasarımı ile ilgilenilmez. Yazılımın fonksiyonelliğini test edebilmek içinse yazılımın çalıştığı ortam ve haberleştiği tüm arayüzlerin simüle edilebilmesi gerekmektedir. Örneğin test edilecek yazılım gömülü bir yazılımsa, aynı ortamı sağlayan bir karta yüklenmeli ve haberleştiği arayüzler gerekleri karşılayacak şekilde oluşturulmalıdır[1].

Yazılımın gömülü bir arayüzü varsa simülatör de buna göre hazırlanmalı aynı dil kullanılarak, aynı işletim sisteminde hatta belki aynı core'da gerçek sistemde karşılık geldiği konfigürasyon birimiyle denk olacak şekilde gerçekleştirilmelidir. Örneklerden de anlaşılabileceği gibi gömülü simülatörlerin kısıtları fazladır. Ayrıca gömülü sistemlerdeki

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

arayüz sayılarının fazla olması nedeniyle testler sırasında kullanılan simülatör sayısı da artmaktadır. Artan simülatör sayısı ile bunların kontrolü ve idamesi de zorlaşmaktadır. Gömülü simülatörlerin test gereklerinden dolayı gömülü bir kart üzerinde koşmasından ötürü mesaj akışını monitör etme ve çalışma sırasında konfigüre edebilme gibi zorlukları da bulunmaktadır. Bunun için de çalışma sırasında gömülü simülatörlerin daha rahat kontrol edilebilmesi için ayrıca bir PC arayüzüne de ihtiyaç duyulmaktadır.

Test sırasında her bir gömülü arayüz için ayrı bir simülatör ve bu gömülü simülatörlerdeki akışı görüntülemek için PC simülatörleri geliştirip her birine akıl yüklemek yerine bütün simülatörlerin ortak bir çatı altında toplanıp tek bir noktadan kontrol ve manipüle edilmesi daha verimli olacaktır. Bundan yola çıkılarak belirtilen ihtiyaçların karşılanması ve simülatör oluşturma için harcanan iş gücünün minumuma indirilmesi amacıyla Ortak Gömülü Simülatör(OGS) geliştirilmiştir.



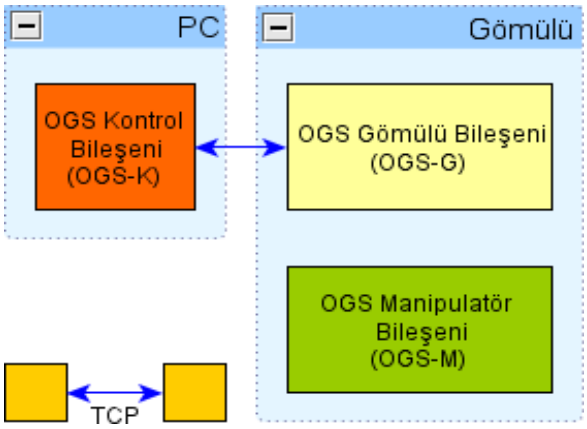
Şekil 1. OGS çalışma şeması. OGS'nin kurabileceği bağlantı arayüzleri görülmektedir. Şekilde PC'deki SIM kısmı ile MsgQ gibi gömülü arayüzlere gelen mesajların yönlendirilip monitör edilebilmesi anlatılmaktadır.

2 Ortak Gömülü Simülatör Nedir?

Ortak Gömülü Simülatör(OGS), farklı cihazlar ve farklı protokoller arasındaki iletişimi ve birbirleri arasında yönlendirmeyi sağlayan bir yazılımdır. Bu işlemler, farklı iletişim protokolleri ile haberleşen gömülü yazılımların OGS gömülü bileşeni ile aynı kart üzerinde farklı çekirdeklere yerleştirilmesi ile gerçekleştirilebilir veya farklı iletişim protokolleri ile haberleşen cihazların OGS gömülü yazılımının çalıştığı kart üzerindeki uygun port girişlerine bağlanmasıyla sağlanabilmektedir. OGS, yönlendirmenin yanı sıra iletişimler arasına girerek kullanıcı tarafından tasarlanan örüntüler ile farklı cihazlar veya yazılımlar arasındaki mesaj trafiğini şekillendirmeye olanak sağlar. Belirlenen koşullar(örüntüler) sağlandığında veya tanımlanan zaman periyoduna göre seçilen mesajın herhangi bir arayüze periyodik olarak gönderimine imkân tanır. Test sırasında senaryo gereği bir seferliğine mesaj tetikleme ve mesajlar için filtre tanımlama gibi yetenekleri bulunmaktadır. Şekil 1'de OGS'nin çalışma şeması görsel üzerinde belirtilmiştir.

OGS iki temel bileşenden oluşur. Bunlar OGS Kontrol (OGS-K) ve OGS Gömülü (OGS-G) bileşenleridir. OGS kontrol bileşeni java ile kodlanmış bir PC yazılımıdır. TCP üzerinden OGS-G bileşeni ile haberleşerek, bu bileşeni çalışma sırasında

yönetir. OGS-G ise gömülü kart üzerinde çalışacak olan C++ ile kodlanmış yazılımdır. OGS deki belirtilen asıl yetenekler bu yazılım üzerinden gerçekleştirilmektedir. Bu bileşenlere ek olarak mesajlar manipüle edilmek istendiğinde, C++ ile kodlanan OGS Manipülatör (OGS-M) bileşeni kullanılmaktadır. OGS bileşenleri Şekil 2'de gösterilmektedir.



Şekil 2. OGS bileşenleri.

3 Ortak Gömülü Simülatör Kurulumu

Kara kutu testi uygulanacak yazılımın arayüz kurması gereken işlemciye OGS-G yüklenir. OGS-G'yi kontrol amacıyla OGS-K yazılımı ile TCP üzerinden bağlantı kurulur ve bu yazılım ile OGS'nin yetenekleri ayarlanarak yazılım testleri için gerekli ortam sağlanmış olur. OGS-K aracı ile OGS-G yazılımının yetenekleri çalışma sırasında değiştirilebilir.

OGS-G tarafından gönderilen mesajlarda çeşitli durumlara göre değişiklik yapılmak isteniyorsa (otomatik cevap gönderilen mesajlarda ID değerinin her seferinde artırılması, her 3 mesajdan birinin hatalı gönderilmesi vs.) işlemciye OGS-G yüklenmeden önce OGS-M yüklenir. Kullanıcılar mesajları OGS-M içerisinde istedikleri şekilde değiştirerek gönderebilirler.

4 Ortak Gömülü Simülatörün Yetenekleri ve Kullanımı

4.1 Arayüzler

OGS ile farklı yazılımların ihtiyaçlarına göre farklı arayüzler tanımlanabilmektedir. OGS'nin desteklediği arayüzler aşağıda verilmiştir:

- UDP
- TCP
- Seri Kanal
- MSGQ
- PCIe

OGS-G'nin bağlanması istenilen arayüzlere ait parametreler OGS-K yazılımı tarafından OGS-G'ye gönderilir. OGS-G bağlanması gereken arayüzleri öğrendiği zaman ilgili arayüz üzerinden yazılım ile haberleşme işlemlerini başlatır.

Tablo 1. Örnek arayüz tanımlama içerikleri.

MessageQueue	Arayüz ID	1
	Protokol	MsgQ
	Istemci Adı	istemci
	Sunucu Adı	sunucu
TCP Server	Arayüz ID	2
	Protokol	TCPServer
	Port	9001
Seri Kanal	Arayüz ID	3
	Protokol	Serial
	Port	COM13
	BaudRate	115200
	ByteSize	8

Arayüzler tanımlanırken Arayüz ID'lerinin birbirlerinden farklı olmasına dikkat edilir. Arayüz bağlantıları tanımlandıktan sonra ilgili arayüzlerden gönderilecek/alınacak mesajların başlık yapısı tanımlanır (bknz. Tablo 1).

Mesaj başlık yapısında bulunan "Byte Order Yapılsın mı" parametresi ile ilgili arayüzün endian formatından farklı bir endian formatına sahip bir kart üzerinde çalışıldığı zaman mesajların doğru bir şekilde parse edilebilmesi sağlanmaktadır.

Tablo 2. Örnek mesaj başlık tanımlama.

Gönderen Alan Location	0
Gönderen Alan Boy	2
Mesaj Id Location	2
Mesaj Id Boy	2
Mesaj Boyu Location	4
Mesaj Boyu Boy	4
Checksum Location	0
Checksum Boy	0
Byte Order Yapılsın mı	0
Arayuz Id	1

Başlık yapıları Arayüz ID'ler kullanılarak gerçekleştirilir (bknz. Tablo 2) ve hatalı tanımlanması durumunda OGS hata vererek çalışmayı durduracaktır. Bundan dolayı tüm tanımlamaların doğru bir şekilde yapılması gerekmektedir.

4.2 Yönlendirme

Gömülü sistemlerde farklı işlevleri gerçeklemek için birden çok yazılım çalışmaktadır. OGS, bu yazılımların birbirleriyle olan haberleşmelerinde kullandıkları mesajları bir arayüzden öteki arayüze yönlendirebilmektedir.

Mesajların OGS tarafından yönlendirilmesinin sebepleri aşağıdaki gibidir:

- PC de oluşturulamayacak arayüzler için OGS'nin PC simülatörü ile gömülü yazılım arasında adaptör görevi görmesi amacıyla yönlendirme tanımlanabilir.
- Arayüzde bulunan mesajları gözlem/analiz etme amacıyla başka araçlara yönlendirme yapılabilir.
- Birden fazla yazılımın kullandığı ortak bilgiler içeren mesajlar da, her yazılım arayüzü için birebir ayarlama yapılması yerine OGS üzerinden tek bir yerden birden fazla yazılıma göndermek üzere tanımlanabilir.

Yönlendirme işlemi 2 şekilde tanımlanabilmektedir. Bunlardan biri arayüzler arasındaki

herhangi bir mesajı yönlendirme diğeri ise herhangi bir arayüze gelen mesajların hepsinin diğer bir arayüze yönlendirilmesidir.

4.2.1 Mesaj Yönlendirme

Bir mesajın bir arayüzden birden fazla arayüze gönderileceği senaryolarda kullanılmak üzere tasarlanmıştır.

Örnek tabloda (bknz. Tablo 3) 2 numaralı arayüzden gelen 1 numaralı mesaj 3 numaralı arayüze gönderilmektedir. Veri İşleme alanının değerine göre ise mesajın değiştirilme durumu belirlenmektedir. Eğer Veri İşleme alanı True ise mesaj 3 numaralı arayüze kullanıcı tarafından daha önce belirtilen değişiklikler uygulanarak gönderilecektir.

Tablo 3. Örnek mesaj yönlendir tanımlama.

Mesaj ID	1
Gelen Arayüz ID	2
Giden Arayüz ID	3
Veri İşleme	True

4.2.2 Bütün Mesajları Yönlendirme

OGS-G'ye gelen mesajların PC üzerindeki bir simülatör de monitör edilebilmesi veya OGS'nin iki arayüz arasında adaptör olarak görev yapabilmesi amacıyla tasarlanmıştır.

Örnek tabloda (bknz. Tablo 4) 2 numaralı arayüzden gelen bütün mesajlar 3 numaralı arayüze gönderilmektedir.

Tablo 4. Örnek hepsini yönlendir tanımlama

Gelen Arayüz ID	2
Giden Arayüz ID	3

4.3 Periyodik Mesaj

Gömülü sistemlerde bakım, kontrol ve yönetim amacıyla yazılımlar arası belirli aralıklarla farklı mesajlar gönderimi/alımı gerçekleşmektedir. OGS ile bu periyodik mesajlar otomatik olarak, tekrar simülatörü derleme ihtiyacı oluşturmadan esnek bir şekilde tanımlanabilmektedir. Periyodik mesajlar, mesaj tetiklemeli, zaman tetiklemeli, mesaj-zaman tetiklemeli olmak üzere üç tür olarak tanımlanabilir.

4.3.1 Mesaj Tetiklemeli Periyodik Mesajlar

Belirli gelen bir mesaja otomatik cevap verilebilmesi amacıyla kullanılmaktadır. Periyot alanı sıfır girilip mesaj id alanı da geçerli bir id girildiğinde, ilgili id'ye sahip mesaj geldiğinde Mesaj alanında tanımlı mesaj gönderilmektedir.

Örnek tabloda (bknz. Tablo 5) 2 numaralı arayüzden 1 numaralı mesaj geldiğinde, 3 numaralı arayüze Mesaj alanında tanımlı olan mesaj gönderilecektir.

Tablo 5. Örnek mesaj tetiklemeli periyodik mesaj tanımlama.

Periyot (ms)	0
Gelen Arayüz ID	2
Giden Arayüz ID	3
Mesaj ID	1
Mesaj	MesajStruct

4.3.2 Zaman Tetiklemeli Periyodik Mesajlar

Belirli periyotlarda mesaj gönderebilmek amacıyla kullanılmaktadır. Periyot alanı sıfırdan farklı bir sayı olarak tanımlandığında yazılım ile haberleşme işlemleri tamamlandıktan sonra gönderilmek istenen mesaj belirlenen periyot ile gönderilmeye başlayacaktır.

Örnek tabloda (bknz. Tablo 6) 5 saniyede bir Mesaj alanında tanımlı olan mesaj 3 numaralı arayüze gönderilmektedir.

Tablo 6. Örnek zaman tetiklemeli periyodik mesaj tanımlama.

Periyot (ms)	5000
Gelen Arayüz ID	0
Giden Arayüz ID	3
Mesaj ID	0
Mesaj	MesajStruct

4.3.3 Mesaj ve Zaman Tetiklemeli Periyodik Mesajlar

Belirli gelen bir mesaja otomatik cevap verebilmek hem de ilgili mesajı periyodik olarak gönderebilmek amacıyla kullanılmaktadır. Periyot alanı sıfırdan farklı ve geçerli bir mesaj id girilerek işlevler yerine getirilebilmektedir.

Örnek tabloda (bknz. Tablo 7) 2 numaralı arayüzden 1 numaralı mesaj geldiğinde 3 numaralı arayüze mesaj gönderilecektir. Ayrıca 5 sn ‘de bir mesaj gönderilmeye devam edecektir.

Tablo 7. Örnek mesaj-zaman tetiklemeli periyodik mesaj tanımlama.

Periyot (ms)	5000
Gelen Arayüz ID	2
Giden Arayüz ID	3
Mesaj ID	1
Mesaj	MesajStruct

4.4 OGS-K Aracından Arayüz Mesajlarını Gönderebilme ve Alabilme

Testler sırasında kullanılan mesajları periyodik olarak tanımlayıp gönderme veya farklı bir simülatörden gönderip yönlendirmek yerine OGS-K aracının direk mesaj gönder özelliği ile mesajlar OGS-K üzerinden de direk olarak gönderilebilmektedir. Bu özellik senaryo gereği bir seferliğine gönderilmek istenilen mesajlar veya basit senaryolar için ayrı bir simülatörün yapılıp OGS-G ile arayüzünün oluşturulması ve yönlendirme işlemlerinin tanımlanması için harcanacak iş gücünü azaltmak amacıyla eklenmiştir.

Tablo 8. Örnek direk mesaj gönderme içeriği.

Giden Arayüz ID	3
Mesaj	MesajStruct

Örnek tabloda(bknz. Tablo 8) ilgili mesaj 3 numaralı arayüze gönderilmektedir.

OGS-K aracından mesaj gönderebilmenin yanı sıra arayüzlerden gelen cevap mesajları OGS-K aracına da gönderilerek mesajların tek bir araç üzerinde monitör edilmesi sağlanmıştır. Bu sayede cevap mesajlarının monitör edilmesi için ayrıca bir simülatör oluşturulup arayüz açılmasına ihtiyaç kalmamıştır.

4.5 Filtre Tanımlama

Arayüzler arası bütün mesajları yönlendirme tanımlama veya OGS-K aracına cevap mesajlarının gönderilmesi özellikleri kullanılırken, periyodik olarak gelip cevaplanan mesajların yönlendirilmesi veya OGS-K aracına gönderilmesinin istenmediği durumlarda

ilgili mesaj için negatif filtre tanımlayarak mesajın yönlendirilmesi ve OGS-K aracına gönderilmesi engellenebilmektedir. Tanımlanan filtre çalışma sırasında güncellenebilmektedir.

Örnek tabloda (bknz. Tablo 9) 2 numaralı arayüzden gelen 1 id'li mesaj için negatif filtre tanımlanmıştır. İlgili arayüz için bütün mesajları yönlendir tanımlanmışsa ilgili mesaj geldiğinde yönlendirilme yapılmayacaktır ayrıca OGS-K aracına da gönderilmeyecektir.

Tablo 9. Örnek negatif filtre tanımlama.

Gelen Arayüz ID	2
Mesaj ID	1
OnOff Durumu	True

4.6 XML Dosyası ile Başlatma

OGS-G yazılımı OGS-K'dan aldığı kontrol mesajları ile yönetilmesi dışında herhangi bir yazılımın testleri için sürekli olarak kullanılabilecek kontrol mesajları(arayüz tanımlama, header tanımlama, periyodik mesaj tanımlama vs.) belirli bir xml dosyası formatında hazırlanarak da OGS başlatılabilmektedir. Test senaryomuz sabit ise bu yetenek sayesinde OGS-K aracından arayüzleri, yönlendirmeleri, periyodik mesajları tanımlamaları her test sırasında yapmak yerine hazırlanan xml dosyası ile bu işlemler tek seferde yapılarak test sırasındaki harcanan iş gücü azaltılmaya çalışılmıştır.

4.7 Mesaj Manipüle Etme

OGS, bir arayüzden mesaj geldiğinde yapılması gereken işlemler (zaman sayacının artırılması, gelen mesajdaki bir değerin cevap mesajında kullanılacak olması, dosyadan veri okuma vs.) için altyapı sunmaktadır. Kullanıcı yapılacak işlemleri tetikleyen mesajın adı ile oluşturduğu C++ dosyası içerisinde tanımlayarak bu dosyayı OGS-M içerisinde işlemciye yüklemelidir. Eğer Yönlendirme mesajlarında ilgili mesajın Veri İşleme alanı True ise kullanıcının tanımladığı işlemler gerçekleştirilecektir.

4.8 Mesaj Log Özelliği

OGS-G bağlandığı log sunucusuna gönderdiği/aldığı tüm mesajların log kayıtlarını göndermektedir. Bu sayede testler sırasında gözden kaçabilecek veri akışları log dosyasından incelenerek testler daha güvenli bir şekilde yapılabilmektedir.

4.9 Tek Noktadan Kontrol

Test edilen yazılımın arayüz sayısı ile doğru orantılı olarak simülatör sayısı da artmaktadır. Artan simülatör sayısı ile beraber simülatörlerin yönetilmesi zorlaşmaktadır, ayrıca PC simülatörlerinin sayısının artması performans sorunlarına da neden olmaktadır. OGS sayesinde OGS-G üzerinde birden fazla simülatör yönetilebilmekte ve gömülü bir sistem olması sayesinde performans sorunları minimuma indirilmiştir. Bu yararlarının yanında OGS-G içerisinde arayüzler birbiri ile kolayca haberleşebilmektedir. Böylece bir arayüze gönderilen mesaj içerisindeki bilgiler, diğer arayüzler tarafından kullanılabilir.

5 Uygulama ve Kazanımlar

OGS REHIS bünyesindeki yeni radar projelerinin kara kutu testlerinde kullanılmaktadır. OGS'nin farklı arayüz bağlantılarını oluşturabilme ve çalışma esnasındaki konfigüre edilebilir yetenekleri sayesinde yeni simülatör ihtiyaçlarında, simülatörlerin sıfırdan kodlanması ihtiyacını ortadan kaldırmıştır. Bu sayede zaman ve maliyet kazancı sağlanmıştır. Ayrıca OGS bileşenlerinin doğrulamaları yapıldığı için kullanıcının yeni kodladığı simülatörün doğrulaması için harcayacağı zaman da ortadan kaldırılmıştır.

OGS, gömülü arayüzler(PCIe, MsgQ) için otomatik test koşturulabilmesinde yardımcı bir araç olarak da kullanılmaktadır. OGS'nin yönlendirme yeteneği ile PC tarafında JUnit vb. framework ler ile senaryolar oluşturulup koşturulabilmektedir. OGS'nin gömülü

arayüzler için otomatik test koşturabilmeye yardımcı olması sayesinde önceki çalışmalarda [2] görüldüğü gibi otomatik testin, manuel teste göre 2 kattan fazla zaman ve maliyet kazancının sağlanmasında rol oynaması hedeflenmektedir.

OGS'nin ilerleyen aşamalarda radar projelerinde tümleşik testlerin yapılabilmesi için donanım tarafının simüle edilmesinde kullanılması planlanmaktadır. OGS'nin tümleşik test altyapılarında kullanılması için çalışmalar yapılmaya devam edilmektedir.

OGS'nin yeni projeler ile birlikte kullanılmaya başlamasından dolayı mevcut durumda performans ve iş gücü kazancı gibi metriklerin ölçümü için yeterli veri yoktur. Kullanıcılar tarafından alınan geri dönüşler ve gözlemler sonucunda testler için harcanan zamanı ve maliyeti düşürdüğü görülmüştür.

6 Sonuç

Bu çalışmada farklı iletişim protokolleri ile haberleşen gömülü yazılımların arasında mesaj alışverişini tek bir noktadan kontrol ve manipüle edebilen, esnek bir şekilde yeni yeteneklerin tanımlanmasına imkân sağlayan OGS yazılımı tanıtılmıştır. OGS sayesinde gömülü yazılım test çalışmalarında gömülü simülatörler tek bir ortak altyapıda toplanarak kullanım kolaylığı, işgücü kazancı ve tekrar kullanımda büyük oranda artış sağlanmıştır.

İlerleyen aşamalarda sık kullanılan işlemler için (dosya işlemleri, işlemci CPU/bellek kullanım analizi vb.) yeni bileşenler tanımlanması hedeflenmektedir.

OGS bu aşamada VxWorks 6.8, 6.9 ve 7.0 işletim sistemlerini desteklemektedir. Bu desteğin gömülü sistemlerde kullanılan işletim sistemlerinin çoğunu kapsayacak şekilde genişletilmesi hedeflenmektedir.

7 Kaynakça

1. Yazılım Testi Metodolojileri Sayfası, http://www.iztim.com/Blog/YazilimTeknolojisi/YAZILIM-METODOLOJI, Son Erişim 19/06/2019.

2. Author, M. Sowers: "Software Testing Tools Summary", White Paper, Software Development Technologies Inc., 2002.

Yazılım Ürün Hatları için Özellik Yönelimli Test Modellerinin Yönetimi

Tuğkan Tuğlular[1], Mutlu Beyazıt[2] ve Dilek Öztürk[1]

[1] Department of Computer Engineering, Izmir Institute of Technology, Izmir, Turkey
tugkantuglular@iyte.edu.tr
dilekozturk@iyte.edu.tr
[2] Department of Computer Engineering, Yaşar University, İzmir, Turkey
mutlu.beyazit@yasar.edu.tr

Özet. Bu çalışma, yazılım ürün hatlarında çok fazla sayıda olabilecek olan özelliklere ve ürünlere ait test modellerinin yönetimini oluşturan süreçler ile ilgili bütünsel bir yaklaşım önermektedir. Bir yazılım ürün hattı özelliklerine ve ürünlerine ilişkin test modellerinin yönetimi; özellik test modeli oluşturma, özellik test modeli doğrulama, ürün için test modellerini birleştirme ve ürün test modeli doğrulama süreçlerinden oluşmaktadır. Bu bildiride, test modellerinin yazılım ürün hatları için özelleştirilmiş olay sıra çizgeleri olduğu bir ortamda, bu modellerin doğrulanmasına ilişkin bir yöntem ile ürün yapılandırması sonucunda ortaya çıkan ürüne ait olay sıra çizge test modelinin doğrulanmasına ilişkin bir yöntem önerilmektedir.

Anahtar Kelimeler: Yazılım Ürün Hatları, Model-tabanlı Test, Olay Sıra Çizgesi, Özellik Modeli.

Management of Feature-Oriented Test Models for Software Product Lines

Abstract. This study suggests a holistic approach for the processes entail in the management of test models belonging to features and products in software product lines, which may be very large in numbers. Management of test models regarding to features and products of a software product line is composed of building feature test models, verification of feature test models, composition of feature test models to obtain product test model, and verification of product test models. In this paper, a method for the verification of feature test models, which are extended event sequence graphs for software product lines, and another method for the verification of product test models, which are event sequence graphs, are proposed.

Keywords: Software Product Lines, Model-based Testing, Event Sequence Graph, Feature Model.

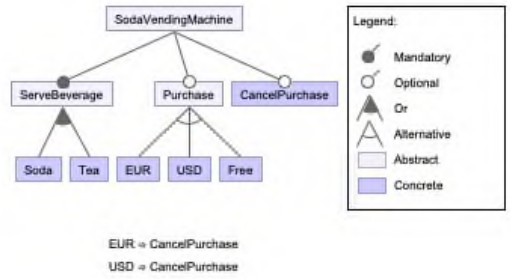
Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

1 Giriş

Yazılım ürün hatlarının (YÜH) model tabanlı test otomasyonu vizyonu içinde önemli parçalardan biri özelliklere ve ürünlere ait test modellerinin yönetimidir. Model yönelimli yazılım mühendisliği; oluşturma, dönüştürme, doğrulama ve birleştirme etkinlikleri aracılığı ile modellerin yönetilmesi ile ilgilenir [1]. Model yönelimli yazılım mühendisliğinde modellerin doğrulanması en zor konulardan biri olarak tanımlanmaktadır [2]. Bu çalışmada, yazılım ürün hatlarında çok fazla sayıda olabilecek olan özelliklere ve ürünlere ait test modellerinin yönetimi hedeflenmiştir. Bir yazılım ürün hattı özelliklerine ve ürünlerine ilişkin test modellerinin yönetimi özellik test modeli doğrulama, ürün için test modellerini birleştirme ve ürün test modeli doğrulama etkinliklerinden oluşmaktadır. Bu çalışmada anılan etkinlerin her biri için bir yaklaşım önerilmektedir.

Yazılım ürün hatları açısından bakıldığında bir özellik (İng. feature), sistem gereksinimlerini karşılayan ve potansiyel yapılandırma seçenekleri sağlayan bir fonksiyonellik birimidir [3]. Bir başka tanıma göre özellik söz konusu yazılımın kullanıcıları için bir anlamı olan ayırt edilebilir yazılım karakteristiğidir [4].

Kang et al. [5] tarafından ortaya atılan özellik şamaları yazılım ürün hatlarında yapılandırma seçeneklerini ve özelliklerin bağımlılıklarını temsil etmek için kullanılmaktadır. Örnek bir YÜH özellik şeması Şekil 1'de verilmektedir. Kök YÜH'ü ve düğümler de zorunlu veya isteğe bağlı özellikleri temsil etmektedir. VEYA (İng. OR) ilişkisi olan özellikler bir üründe farklı birleşimlerde bulunabilir. İki özellik arasında DIŞLAYAN VEYA (İng. XOR) ilişkisi varsa bunlardan yalnızca biri YÜH'te yer alabilir. Özellik şemalarında aynı zamanda GEREKTİRME ve DIŞLAMA ilişkileri de bulunabilir. Burada ilk ilişki bir özelliğin gerektirdiği diğer özellik olmadan olamayacağını, ikincisi ise özelliğin dışlanan bir diğerini göz ardı ettiğini göstermektedir.



Şekil 1. FeatureIDE [6] Kullanılarak Oluşturulan İçecek Otomatı Yazılım Hattı Özellik Şeması [8] ([7]'den değiştirilmiştir)

Özellik şemaları genellikle kullanıcı merkezlidir. Özellik modelleri, özellik şemalarının biçimsel gösterimleridir. Ürün şemaları ise, ürün için seçenekler ve seçimlerle ilgili tüm kararların alındığı ve gerekli bağımlılıkların somutlaştırıldığı ürün yapılandırmasının kullanıcı merkezli gösterimleridir. Bu bildiride, [8]'de yayınlanan çalışmanın devamı olan adımlar yer almaktadır. Tutarlılık açısından [8]'de kullanılan örnekler

aynen alınmıştır. Tablo 1'de soda otomatı YÜH için ürün yapılandırmaları verilmektedir. Bu tabloda USD-Soda-Tea adı verilen ürün için ServeSodaBeverage, ServeTeaBeverage, CancelPurchase ve USDPurchase özelliklerini göstermektedir.

Tablo 1. FeatureIDE Kullanılarak Oluşturulan İçecek Otomatı Yazılım Hattı Ürün Yapılandırmalarına İlişkin Özellik Seçimleri



İşlevsel test hedefine yönelik test modelleri genellikle UML etkinlik diyagramları, sınırlı durum makinaları gibi davranışsal modellerden oluşturulur. Bu çalışmada test modeli olarak yazılım ürün hatları için özelleştirilmiş olay sıra çizgeleri (OSÇ) kullanılmıştır [8]. Yazılım ürün hatları için özelleştirilmiş OSÇ'lerin çizilmesi için web tabanlı bir uygulama geliştirilmiştir [9]. Bu çalışmada ise, YÜH'lerinde kullanılan test modellerinin yönetimi için yeni bir yaklaşım ortaya konulmuştur.

İzleyen kısımda önerilen iki yöntem açıklanmıştır. Bir sonraki kısımda YÜH'lerde test modellerinin yönetimi süreci etkinlik bazında açıklanmış ve bütünsel bir yaklaşım sunulmuştur. Örnek çalışma kısmında ise örnek bir YÜH için önerilen yöntemlerin çalışma zamanları ortaya konmuştur. Sonuç bölümünde önerilen yöntemlerin ve bütünsel yaklaşımın önemi vurgulanmıştır.

2 Özellikli Olay Sıra Çizgelerinin Doğrulanması

Bu kısımda YÜH'lerde OSÇ'leri kullanarak modelleme ve oluşturulan modellerin doğrulanmasıyla ilgili önerilen yaklaşım anlatılmaktadır.

2.1 Temel Tanımlar

Bu çalışmada aşağıdaki özellik modeli ve ilgili ürün yapılandırma, yani özellik seçimi, tanımları kullanılmaktadır.

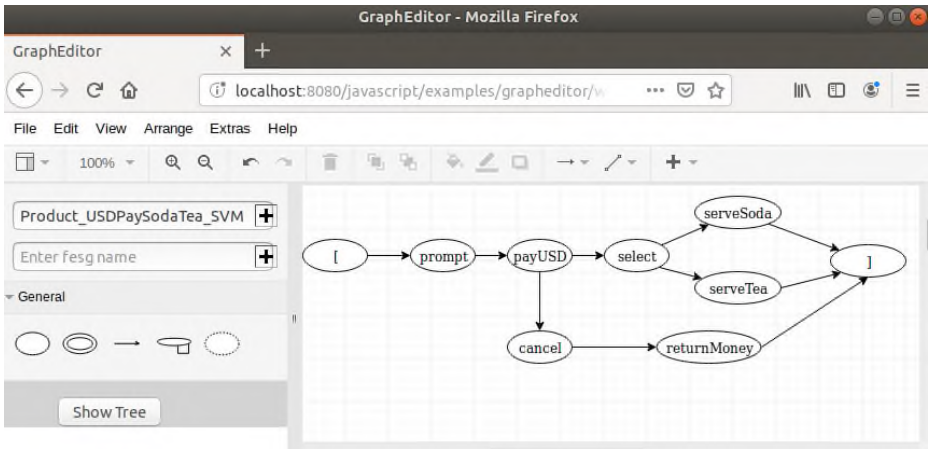
Tanım 1. B = {yanlış, doğru}, Boole değerlerini içeren tanım kümesi ve F, Boole değişkenleri içeren sonlu bir küme olsun. F sınırlı bir Boole değişkenleri kümesi (özellikler) olsun. Bir özellik modeli (ÖM) fm: (F → B) → B, F kümesi üzerinde tanımlı bir önerme formülü olarak verilir [10]. ■

Tanım 2. Bir ürün yapılandırması (ÜY) pc: F → B, özelliklere fm(p) = doğru eşitliğinin sağlayacak şekilde Boole değerlerinin atanmasıdır [10]. ■

Bu çalışmada YÜH'ler üzerinde model tabanlı bir yaklaşım geliştirmek için sadelik ve kullanım kolaylığından dolayı olay sıra çizgeleri kullanılmaktadır.

Tanım 3. Bir olay sıra çizgesi (OSÇ) (İng. event sequence graph - ESG) (V,E) bir yönlü çizgedir; öyle ki V ≠ ∅ sonlu bir olaylar (düğümler ya da köşeler) kümesi ve E ⊆ V×V sonlu bir yaylar (kenarlar) kümesidir. [ve] sırasıyla başlangıç ve bitiş olaylarını işaretlemek için kullanılan sözde olaylardır; [olayına gelen yay ve] olayından giden yay bulunmamaktadır [11]. ■

Şekil 2 ile verilen OSÇ için, V = {[, prompt, payUSD, select, serveSoda, serveTea, cancel, returnMoney,]} ve E = {([, prompt), (prompt, payUSD), (payUSD, select), (payUSD, cancel), (select, serveSoda), (select, serveTea), (cancel, returnMoney), (serveSoda,]), (serveTea,]), (returnMoney,])}. Bu OSÇ'de prompt başlangıç olayı iken serveSoda, serveTea ve returnMoney bitiş olaylarıdır.



Şekil 2. Web Tabanlı Özellikli Olay Sıra Çizge Aracı [9] İçinde USD-Soda-Tea İçin Ürün OSÇ'si [8]

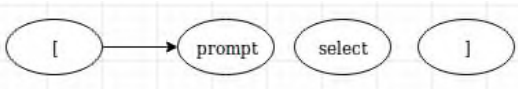
Yazılım ürün hatları için özelleştirilmiş OSÇ'ler çekirdek OSÇ (ç-OSÇ) ve özellik OSÇ (ö-OSÇ) olmak üzere iki tiptir [8].

Tanım 4. Bir özellikli olay sıra çizgesi (ÖOSÇ) (İng. Featured event sequence graph - FESG), bir çekirdek OSÇ'den (ç-OSÇ) (İng. core-ESG) ve ÜY bilgisine dayalı bir küme özellik OSÇ'den (ö-OSÇ) (İng. feature-ESG) oluşur [8]. ■

Bir ç-OSÇ YÜH'ü üzerindeki tüm ürün yapılandırmalarında bulunan temel özellikleri betimlerken ö-OSÇ'ler özellik modelindeki özellikleri betimler. Sıradan OSÇ düğümlerinden ve ç-OSÇ düğümlerinden farklı olarak bir ö-OSÇ bağlantı olayları olarak

adlandırılan değişkenlik noktalarıyla ilişkilendirilmiş düğümleri içerir. Bağlantı olayları aslında ç-OSÇ veya diğer ö-OSÇ'lerdeki olaylardır. Bu olaylar, (OSÇ, Olay) çiftleri olarak gösterilir. Bu nedenle, bir ÖOSÇ bir OSÇ'ye dönüştürülebilir; ancak, bunun tersi ek bilgi kullanılmadan mümkün değildir. Ayrıca, bir ÖOSÇ'deki sözde olaylar ve bağlantı olayları dışındaki olayların hepsi eşsiz olarak adlandırılmalıdır.

Şekil 3'de örnek soda otomatı YÜH'ü için ç-OSÇ verilmektedir. prompt ve select olayları soda otomatı YÜH'ünün çekirdek davranışı içerisindedir ve soda otomatı YÜH'ünden üretilen tüm ürünlerde bulunmaktadır. Belirli bir YÜH ürününün davranışını oluşturmak için ö-OSÇ ile gösterilen seçili özelliklerin davranışları, ç-OSÇ'ye bağlanır. Bağlantı bilgisi ö-OSÇ'lerde saklanmaktadır [8].



Şekil 3. Soda Otomatı YÜH'ü İçin ç-OSÇ [8]

Şekil 4 soda otomatı YÜH'ünün SeveTeaBeverage özelliğinin davranışını göstermektedir. Bu ö-OSÇ, serveTea olayının bağlanabileceği en soldaki düğümde ve en sağdaki düğümde görülen bağlantı noktası bilgilerini içerir. Şekil 5 payUSD olayının Core_SVM ç-OSÇ'ye ve ayrıca Feature_Cancel ö-OSÇ'ye bağlanacağı soda otomatı YÜH'ünün USDPurchase özelliğinin davranışını göstermektedir. Bu bağlantılar zorunludur.



Şekil 4. Soda Otomatı YÜH'ünün ServeTeaBeverage Özelliğine Ait ö-OSÇ [8]



Şekil 5. Soda Otomatı YÜH'ünün USDPurchase Özelliğine Ait ö-OSÇ [8]

Bir ç-OSÇ'ye yeniden kullanımdan yararlanmak için bağlantı olayları dahil edilmemektedir. Bununla birlikte, her ö-OSÇ bağlanacağı OSÇ'ye dair bağlantı olayları, yani değişkenlik noktaları hakkında gerekli bilgileri tutmaktadır. Ürün şemasındaki her yaprak için bir ö-OSÇ vardır.

Bir ürüne ait bir ç-OSÇ ve bir küme ö-OSÇ birlikte ürüne ait bir ÖOSÇ'yi oluşturur. Bu model olası tüm özelliklerin üst üste bindirilmesiyle oluşturulan bir model değildir.

2.2 OSÇ'lerin Doğruluğu

Uygulamada bir model ve öğelerinin tamamıyla kullanılabilirliğini garantilemek gerekebilir. Aksi takdirde, bazı model öğeleri tarafından betimlenen davranışlar doğru şekilde incelenemeyebilir. Bu sebeple kullanışlılık (İng. usefulness) kavramının tanımlanması önem taşımaktadır.

Tanım 5. G = (V, E) bir OSÇ olsun. G'ye ait e ∈ E olayı için eğer e olayına sözde başlangıç olayı ['dan ulaşılabiliyorsa ve e olayından sözde bitiş olayı]'na ulaşılabiliyorsa e olayı G'de kullanışlıdır. Eğer G'deki tüm olaylar kullanışlı ise G kullanışlıdır. ■

Doğru bir OSÇ (İng. valid ESG) kavramı ilk olarak Belli ve Budnik tarafından ortaya atılmıştır [12]. Bu çalışmada ise, doğru bir OSÇ modelinden kastedilen, Tanım 6'de ifade edildiği şekilde kullanışlı bir OSÇ olmasıdır. Doğru OSÇ'lerin bu şekilde tanımlanması OSÇ modellerinin ilgili bağlamlarda etkin bir şekilde kullanılabilmesine olanak vermektedir. Verilen bir OSÇ'nin kullanışlı olup olmadığının denetimi farklı şekillerde yapılabilir. Bunlardan ikisi aşağıda örneklenmiştir:

1. OSÇ'deki her bir e olayı için [olayından e olayına ulaşıp ulaşılmadığı ve e olayından] olayına ulaşılıp ulaşılmadığı kontrol edilebilir.
2. [olayından tüm olaylara ulaşılıp ulaşılmadığı ve] olayından ters yöndeki yayları izleyerek tüm olaylara ulaşılıp ulaşılmadığı kontrol edilebilir.

İki yaklaşım da genişlik yönelimli arama algoritması benzeri bir algoritma kullanarak gerçeklenebilir. 2 numaralı yaklaşım üzerinden kullanışlılık denetimi yapan algoritma Algoritma 1'de verilmektedir.

Algoritma 1. OSÇ Kullanışlılık Denetimi

```
Girdi: G=(V,E) – bir OSÇ
Çıktı: true – G kullanışlı ise, false – G kullanışsız ise
   Q ← boş bir kuyruk yarat
   F ← V'deki düğümlerle indekslenen ve tüm değerleri false olan bir boolean dizisi yarat
   Q'ya ['yi koy, F['['] ← true, c ← 1 //[ ile başla, [ için "ulaşıldı" işaretle, Ulaşılan düğümlerin
                                                    sayısı c'yi 1 yap
   while Q boş değilse do //Genişlik yönelimli olarak tüm düğümlere ulaşmaya çalış
      e ← Q'dan bir eleman al
      for (e,x) ∈ E with F[x] = false do Q'ya x'i koy, F[x] ← true, c ← c + 1 endfor
   endwhile
   if c = |V| then //Tüm düğümlere [ olayından ulaşıldı ise
      Q'yu ve F'yi sıfırla
      Q'ya ]'yi koy, F[']'] ← true, c ← 1 //] ile başla, ] için "ulaşıldı" işaretle, Ulaşılan
                                                    düğümlerin sayısı c'yi 1 yap
      while Q boş değilse do //Genişlik yönelimli olarak tersten tüm düğümlere ulaşmaya çalış
         e ← Q'dan bir eleman al
         for (x,e) ∈ E with F[x] = false do Q'ya x'i koy, F[x] ← true, c ← c + 1 endfor
      endwhile
      if c = |V| then //Tüm düğümlere ] olayından tersine giderek ulaşıldı ise
         return true
      endif
   endif
   return false
```

Şekil 2'de verilen OSÇ modeli doğru bir modeldir; modeldeki tüm olaylara sözde başlangıç olayından ve tüm olaylardan sözde bitiş olayına ulaşılabilmektedir. Diğer taraftan, Şekil 3'te verilen ç-OSÇ, Şekil 4'te verilen ö-OSÇ ve Şekil 5'te verilen ö-OSÇ

modelleri doğru OSÇ modelleri değildir. Özellikleri temsil eden bu OSÇ'lerin doğruluk denetimi izleyen bölümde açıklanmaktadır.

2.3 ÖOSÇ'lerin Doğruluğu

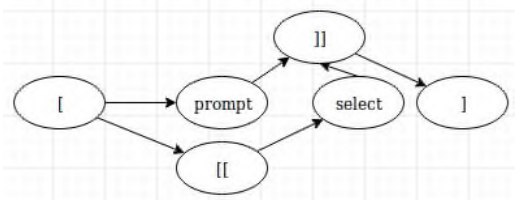
Kısım 2.2 sonunda örneklendiği gibi, ÖOSÇ tabanlı modellemede kullanılan ç-OSÇ ve ö-OSÇ modellerinin kullanışlı olmaması olası bir durumdur. Zira bu modeller beraberce bir ürüne ait tam bir davranışı betimlemektedir. Bu sebeple, asıl önemli olan bu modellerin birlikte kullanışlı bir model oluşturup oluşturmadığıdır.

Yine de modellerin belli bağlamlarda kullanımı (sınama kümeleri üretme vb.) sırasında kolaylık sağlaması için ç-OSÇ ve ö-OSÇ modellerinin de tek başlarına kullanışlı (dolayısıyla doğru) olması faydalı olabilmektedir. Bu modelleri, kullanışlı modellere dönüştürmek amacıyla başlangıç ve bitiş işaretleme sözde olaylarının (sırasıyla [ve]) yanı sıra başlangıçtan kullanışlılık ve bitişten kullanışlılık sözde olayları da (sırasıyla [[ve]]) gerekmektedir.

[[ve]] olaylarının [ve] olayları ile beraber kullanımı aşağıdaki koşullar çerçevesinde olmalıdır:

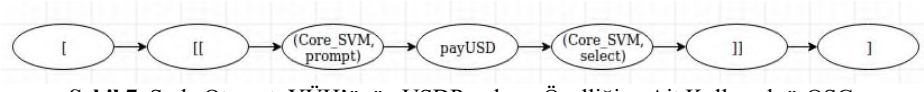
1. [[olayından önce kesinlikle [olayı gelmelidir ve]] olayından sonra kesinlikle] olayı gelmelidir.
2. [[olayından sonra gelen olaylar başlangıçtan ulaşılamayan ve kendisinden önce başka bir olay gelmeyen kullanışsız olaylar olmalıdır ve]] olayından önce gelen olaylar kendisinden bitişe ulaşılamayan ve kendisinden sonra başka olay gelmeyen kullanışsız olaylar olmalıdır.

Şekil 6'da Şekil 3'te verilen ç-OSÇ'nin kullanışlı hali görülmektedir. ç-OSÇ içerisinde hali hazırda [ve] olayları kullanıldığından sadece [[ve]] doğru şekilde eklenerek model kullanışlı hale getirilmiştir. Şekil 3'te verilen ç-OSÇ'deki select olayı kullanışlı değildir; hem [olayından bu olaya ulaşmak mümkün değildir hem de bu olaydan] olayına ulaşmak mümkün değildir. Bunun mümkün hale getirmek adına OSÇ modeline {([, [[), ([[, select), (select,]]), (]],])} yayları eklenmiştir.



Şekil 6. Soda Otomatı YÜH'ü İçin Kullanışlı ç-OSÇ

Şekil 7'de de Şekil 5'te verilen ö-OSÇ'nin kullanışlı hali görülmektedir. ö-OSÇ'deki olayları kullanışlı hale getirmek için [[olayı (Core_SVM, prompt) olayına bağlanmıştır; çünkü, bu olaydan OSÇ'deki diğer tüm olaylara ulaşılabilmektedir. Ayrıca,]] olayı (Core_SVM, select) olayına bağlanmıştır; çünkü OSÇ'deki diğer tüm olaylardan bu olaya ulaşılabilmektedir. Sonuç olarak {([, [[), ([[, (Core_SVM, prompt)), ((Core_SVM, select),]]), (]],])} kümesindeki yaylar OSÇ'ye eklenmiştir.



Şekil 7. Soda Otomatı YÜH'ünün USDPurchase Özelliğine Ait Kullanışlı ö-OSÇ

Bu kısmın başında bahsedildiği gibi bir ç-OSÇ ve ö-OSÇ'lerden oluşan bir ÖOSÇ'nin doğruluğu bu modellerin birlikte kullanışlı bir model oluşturup oluşturmadığıyla ilgilidir. Bu denetimin yapılabileceği iki farklı yaklaşım aşağıda verilmektedir:

1. ÖOSÇ modeli içerisindeki ç-OSÇ ve ö-OSÇ modellerinin birleşimi ile ürüne ait bir OSÇ modeli oluşturup bu modelin kullanışlılığı Kısım 2.2'de anlatıldığı gibi kontrol edilebilir.
2. ç-OSÇ'ye ait [olayından ÖOSÇ modelinde yer alan OSÇ'lerdeki tüm olaylara ulaşılıp ulaşılmadığı ve ç-OSÇ'ye] olayından ters yöndeki yayları izleyerek ÖOSÇ modelinde yer alan OSÇ'lerdeki tüm olaylara ulaşılıp ulaşılmadığı kontrol edilebilir.

Algoritma 2'de 1 numaralı yaklaşım uygulanarak ÖOSÇ'ler için kullanışlılık denetimi yapan bir algoritma yer almaktadır.

Algoritma 2. ÖOSÇ Kullanışlılık Denetimi

Girdi: $G=((V_c,E_c), \{(V_{f1},E_{f1}),(V_{f2},E_{f2}),\ldots,(V_{fk},E_{fk})\})$ – Bir ç-OSÇ ve bir küme ö-OSÇ'den oluşan bir ÖOSÇ
Çıktı: true – G kullanışlı ise, false – G kullanışsız ise
  (V,E) ← boş bir OSÇ yarat
  V ← V ∪ {[,]}
  for (V_i,E_i) G'deki bir OSÇ ise **do**
    for $v \in V_i \setminus \{[,],[[,]]\}$ **and** v bir bağlantı olayı değilse **do** V ← V ∪ {v} **endfor**
  endfor
  for (V_i,E_i) G'deki bir OSÇ ise **do**
    for $(x,y) \in E_i$ **with** $x \notin \{[[,]]$ **and** $y \notin \{[[,]]\}$ **do**
      if x bir bağlantı olayı ise yani x = (a,b) **then** x' ← b **else** x' ← x **endif**
      if y bir bağlantı olayı ise yani y = (a,b) **then** y' ← b **else** y' ← y **endif**
      E ← E ∪ {(x',y')}
    endfor
  endfor
  z ← (V,E) OSÇ'siyle Algoritma 1'i işlet
  return z

3 Özellikli Olay Sıra Çizge Test Modellerinin Yönetimi

Bir yazılım ürün hattı özelliklerine ve ürünlerine ilişkin test modellerinin yönetimi aşağıda verilen etkinliklerden oluşmaktadır:

- Özellikli olay sıra çizge test modellerinin oluşturulması ve güncellenmesi
- Özellikli olay sıra çizge test modelinin doğrulanması
- Ürün için test modellerinin birleştirilmesi
- Ürün test modelinin doğrulanması

Anılan test modelleri oluşturulmadan önce YÜH'e ilişkin özellik modelinin ve ürün

yapılandırmalarının hazır olması gereklidir. Özellik modeline ilişkin bir örnek Şekil 1'de, ilgili ürün yapılandırmalarına ilişkin bir örnek ise Tablo 1'de verilmişti. Özellik modeli ile ürün yapılandırmalarının yönetimi de YÜH'lerde önemli bir konudur. Bu alanda en gelişmiş araçlardan biri Eclipse Modeling Framework [13] üzerinde çalışan FeatureIDE aracıdır. Bu araç 2005 yılında geliştirilmeye başlanmış [14] ve halen geliştirilmesi devam etmektedir [15]. Bu çalışmada da özellik modeli ve ürün yapılandırmalarının tanımlanması için FeatureIDE aracından yararlanılmıştır. FeatureIDE, özellik modelini bir XML dosyasında, ürün yapılandırmalarını ise ayrı ayrı metin dosyalarında tutmaktadır.

YÜH için özelleştirilmiş OSÇ'lerin çizilmesi, diğer bir deyişle test modellerinin oluşturulması, için web tabanlı bir uygulama geliştirilmiştir [9]. Bu web tabanlı uygulama, OSÇ'leri ve özelleştirilmiş OSÇ'leri JSON formatında tutmakta ve MongoDB veritabanında saklamaktadır. Bu web tabanlı uygulama, anılan test modellerinden sınama durumlarının üreten ve bildirinin yazarları tarafından geliştirilen bir test üretim makinası (İng. test generation engine) ile birlikte çalışmaktadır. Bu test üretim makinası hem kendi başına çalışabilecek hem de web tabanlı uygulama ile birlikte çalışabilecek şekilde tasarlanmıştır.

FeatureIDE'de üretilen özellik modeli ve ürün yapılandırmalarının test üretim makinası tarafından kullanılabilmesi için XML ve metin dosyaları JSON formatına dönüştürülmektedir. Bu sayede FeatureIDE'de üretilen özellik modeli ve ürün yapılandırmalarının tarafımızdan geliştirilen test üretim makinası tarafından tüketilmesi mümkün olmaktadır. Böylece, bir YÜH için özellik modeli ve ürün yapılandırmaları yönetimi ile test modelleri yönetimi birlikte çalışabilmektedir.

Geliştirilen web tabanlı uygulama sayesinde, test modelleri yönetiminin ilk adımı olan özellikli olay sıra çizge test modellerinin oluşturulması ve güncellenmesi etkinliği sağlanmaktadır. İkinci adım olan özellikli olay sıra çizge test modelinin doğrulanması etkinliği için Kısım 2.3'de önerilen yöntem kodlanmış ve çalışır hale getirilmiştir.

YÜH test modelleri yönetiminin üçüncü adımı olan ürün için test modellerini birleştirilmesi iki şekilde gerçekleştirilebilir. İlk yaklaşımda, tüm özellikler bir araya getirilerek ürün test modeli elde edilir. İkinci yaklaşımda ise, var olan bir ürün test modelinden yola çıkılarak hedeflenen üründeki eksik özellikler eklenerek hedeflenen ürüne ait test modeline ulaşılır. Her iki yaklaşımı da karşılayan önerimiz [8]'de açıklanmıştır.

Ürün test modeli doğrulanması etkinliği YÜH test modelleri yönetiminin son adımıdır. Bu etkinlik için Kısım 2.2'de önerilen yöntem kodlanmış ve çalışır hale getirilmiştir. Böylece, YÜH test modellerinin uçtan uca yönetimine ilişkin tüm etkinlikler için yöntem ve araç oluşturulmuştur. İzleyen kısımda bir örnek üzerinde geliştirilen kodların çalıştırılması ve sonuçları irdelenmiştir.

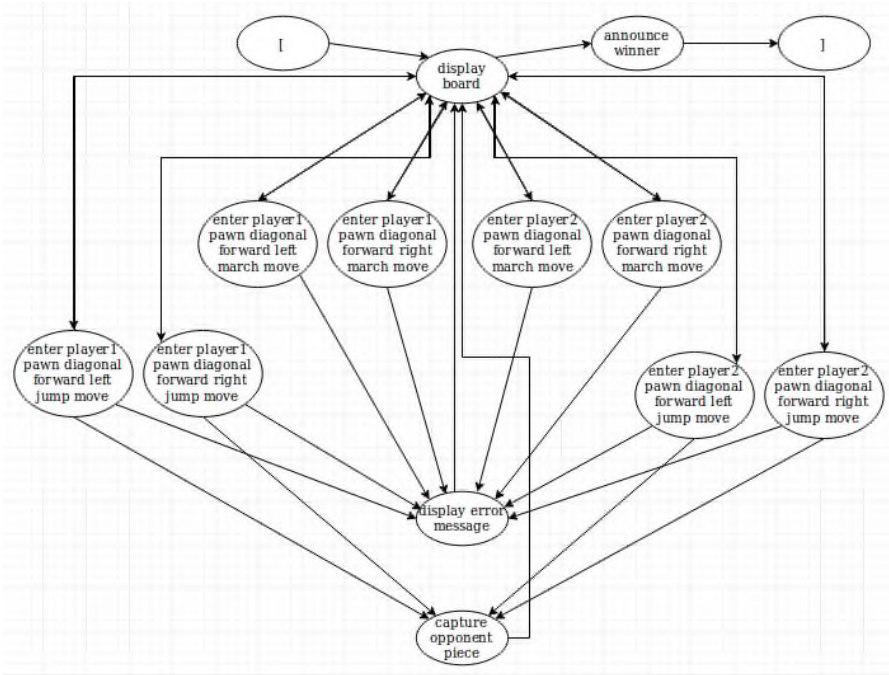
4 Örnek Çalışma

Bu kısımda önerilen yöntemleri açıklarken kullanılan içecek otomatı örneğinden farklı olarak [8]'de gösterilen Amerikan ya da İngiliz dama ürün ailesi YÜH'ü örnek çalışma olarak kullanılacaktır. En kolay dama türü olduğu için seçilen bu YÜH'den; Çocuk daması, Amerikan daması ve İspanyol daması olmak üzere üç yazılım ürünü çıkarılması

hedeflenmiştir.

Amerikan daması 12 adet piyon ile satranç tahtasında oynanır. Bir piyon çapraz hareket eder ve rakibin taşının üzerine atlayarak onu ele geçirir. Piyonlar yalnızca son satıra kadar iler. Son satıra ulaştıktan sonra piyonlar vezire terfi eder ve ileri ya da geri hareket edebilir. Bir oyuncunun tüm parçaları ele geçirildiğinde oyun sona erer. İspanyol damasında piyonların kraliçelere terfi etmesi dışında tüm kurallar aynıdır. Her iki damada da imkan varsa bir taş rakibin taşını almak zorundadır. Bu kural çocuk damasında bulunmamaktadır. Ayrıca, Çocuk damasında bir piyon son sıraya ulaştığında oyun sona erer [8].

Amerikan dama ürün ailesi YÜH'üne ait ÖOSÇ'lerin her biri oluşturulduktan sonra özellikli olay sıra çizge test modeli doğrulanmasından geçer. Böylece, herhangi bir ürün yapılandırması sonucunda ortaya çıkacak ürüne ait test modeli OSÇ'nin özellik test modellerinden gelen bir hatayı barındırmayacağından emin olunur. Şekil 8'de çocuk daması ürününe ait OSÇ görülmektedir. Bu OSÇ, Amerikan dama ürün ailesi YÜH'üne ait ÖOSÇ'lerden bir araya getirilmiştir. Diğer ürünlere ait OSÇ'lerin düğüm ve yay sayıları Tablo 2'de verilmiştir.



Şekil 8. Çocuk Daması Ürünü OSÇ'si [8]

Tablo 2. Amerikan Dama Ürün Ailesi YÜH'üne ait OSÇ'lerin Düğüm ve Yay Sayıları [8]

	Düğüm Sayısı	Yay Sayısı
Çocuk Daması Ürününe ait OSÇ	14	33
Amerikan Daması Ürününe ait OSÇ	31	95
İspanyol Daması Ürününe ait OSÇ	31	95

Bu bildiride önerilen ÖOSÇ'lerin doğruluğu ve OSÇ'lerin doğruluğu denetim yöntemlerinin seçilen örnek YÜH üzerinde işletilmesi sonucunda elde edilen sonuçlar Tablo 3'de ve Tablo 4'de verilmiştir. Tablo 3'de ve Tablo 4'de yer alan değerler ilgili doğruluk denetim kodların birbirinden bağımsız olarak onar defa çalıştırılmaları sonucunda ortaya çıkmıştır. Bu değerler Linux Ubuntu 18.04 işletim sisteminde Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz işlemcili bilgisayar üzerinde elde edilmiştir.

Tablo 3. Amerikan Dama Ürün Ailesi YÜH'üne ait ÖOSÇ'lerin Doğruluk Denetim Zamanına İlişkin Ölçümler (milisaniye)

	Asgari	Azami	Ortalama
Çocuk Daması Ürününe ait ÖOSÇ'ler	5	6	5.2
Amerikan Daması Ürününe ait ÖOSÇ'ler	7	9	7.7
İspanyol Daması Ürününe ait ÖOSÇ'ler	7	11	8.3

Tablo 4. Amerikan Dama Ürün Ailesi YÜH'üne ait OSÇ'lerin Doğruluk Denetim Zamanına İlişkin Ölçümler (milisaniye)

	Asgari	Azami	Ortalama
Çocuk Daması Ürününe ait OSÇ	3	5	3.8
Amerikan Daması Ürününe ait OSÇ	4	7	5.1
İspanyol Daması Ürününe ait OSÇ	4	7	5.3

5 Sonuç

Günümüzde özellikle platform temelli yazılın üretim hatlarının artması ile bu üretim hatlarında oluşturulan ürünlerin model tabanlı testleri giderek önem kazanmaktadır. Bu önemi doğrudan etkileyen süreç ise ortaya çıkan test modellerinin yönetimidir. Bu çalışmada YÜH'ler için test modellerinin yönetimine ilişkin bir yaklaşım önerilmiştir. Bu yaklaşımda test modellerinin yönetimi, dört ana etkinliğe bölünmüş ve her bir etkinliğe ilişkin bir çözüm öneri sunulmuştur. Bunlardan iki tanesi, bildiri yazarlarının önceki çalışmalarından alıntılanarak anlatılırken, diğer ikisi olan özellikli olay sıra çizge test modelinin doğrulanması ile ürün test modelinin doğrulanması etkinliklerine ilişkin ise bu bildiride birer yöntem önerilmiştir. Önerilen yöntemler bir örnek çalışma üzerinde çalıştırılmış ve elde edilen sonuçlar paylaşılmıştır.

Teşekkür

Bu çalışma, TÜBİTAK tarafından 117E884 numaralı "Çevik Yazılım Ürün Hatları için Olay Sıra Çizge Tabanlı Test Üretim Yöntemi Geliştirilmesi" başlıklı proje kapsamında desteklenmiştir, teşekkürlerimizi sunarız.

Kaynakça

1. García-Domínguez, A., Kolovos, D.S., Rose, L.M., Paige, R.F., Medina-Bulo, I.: EUnit: a unit testing framework for model management tasks. In: International Conference on Model Driven Engineering Languages and Systems. pp. 395–409. Springer (2011).

2. Van Der Straeten, R., Mens, T., Van Baelen, S.: Challenges in model-driven software engineering. In: International Conference on Model Driven Engineering Languages and Systems. pp. 35–47. Springer (2008).
3. Apel, S., Kästner, C.: An overview of feature-oriented software development. Journal of Object Technology. 8, 49–84 (2009).
4. Czarnecki, K.: Ulrich Eisenecker Generative Programming: Methods, Tools, and Applications. Addison-Wesley, Paperback, Published June (2000).
5. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-oriented domain analysis (FODA) feasibility study. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst (1990).
6. Thüm, T., Kästner, C., Benduhn, F., Meinicke, J., Saake, G., Leich, T.: FeatureIDE: An extensible framework for feature-oriented software development. Science of Computer Programming. 79, 70–85 (2014).
7. Classen, A.: Modelling and model checking variability-intensive systems. Ph. D. dissertation (2011).
8. Tuglular, T., Beyazıt, M., Öztürk, D.: Featured Event Sequence Graphs for Model-Based Incremental Testing of Software Product Lines. Proceedings of the 43rd International Conference on Computer, Software and Applications (COMPSAC 2019), Milwaukee, Wisconsin, USA (2019).
9. Tuglular, T., Beyazıt, M., Öztürk, D., Demirci, T., Erdoğan, B.: Development of a Web-based Test Generation Tool for Agile Software Product Lines. Proceedings of the 5th Workshop on Dependability at Izmir Institute of Technology, Izmir, Turkey (2019).
10. Lochau, M., Mennicke, S., Baller, H., Ribbeck, L.: Incremental model checking of delta-oriented software product lines. Journal of Logical and Algebraic Methods in Programming. 85, 245–267 (2016).
11. Belli, F.: Finite state testing and analysis of graphical user interfaces. Proceedings of the 12th International Symposium on Software Reliability Engineering, 2001. ISSRE 2001. pp. 34–43. IEEE, Washington, DC (2001).
12. Belli, F., Budnik, C.J.: Minimal spanning set for coverage testing of interactive systems. International Colloquium on Theoretical Aspects of Computing. pp. 220–234. Springer (2004).
13. Steinberg, D., Budinsky, F., Merks, E., Paternostro, M.: EMF: eclipse modeling framework. Pearson Education (2008).
14. Leich, T., Apel, S., Marnitz, L., Saake, G.: Tool support for feature-oriented software development: featureIDE: an Eclipse-based approach. Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange. pp. 55–59. ACM (2005).
15. Meinicke, J., Thüm, T., Schröter, R., Benduhn, F., Leich, T., Saake, G.: Mastering Software Variability with FeatureIDE. Springer (2017).

Pratikte Model Tabanlı Test: Bankacılık Alanından Bir Deneyim Raporu

Şerafettin Şentürk[1], Abdurrahman Akın[1], Ayşe Betül Karagöz[1], Vahid Garousi[2]
Kuveyt Türk R&D Center Kocaeli, Türkiye[1]
{serafettin.senturk, abdurrahman.akin, ayse.karagoz}@kuveytturk.com.tr
Queen's Üniversitesi Belfast Belfast, Birleşik Krallık[2]
v.garousi@qub.ac.uk

Özet. Model odaklı yazılım mühendisliği son yıllarda daha popüler hale geldi. Kullanıcı sayısının ve çeşitliliğinin yüksek olmasından dolayı, hata oluşumunu azaltmak ve endüstriyel uygulamalarda daha yüksek kalite sağlamak için yeni test yaklaşımları gereklidir. Bu çalışmanın amacı, Türkiye'deki büyük bir bankacılık kurumu bağlamında internet bankacılığı çözümlerinin doğrulaması için otomatik testlerin geliştirilmesinde ve koşumunda Model Tabanlı Test (MBT) deneyimlenmesidir.
Anahtar Kelimeler: Model Tabanlı Test, Graphwalker, İnternet Bankacılığı, Deneyim Raporu, Endüstriyel Vaka Çalışması

Model Based Testing In Practice: An Experience Report From The Banking Domain

Şerafettin Şentürk[1], Abdurrahman Akın[1], Ayşe Betül Karagöz[1], Vahid Garousi[2]
Kuveyt Türk R&D Center Kocaeli, Turkey[1]
{serafettin.senturk, abdurrahman.akin, ayse.karagoz}@kuveytturk.com.tr
Queen's University Belfast, Belfast, UK[2]
v.garousi@qub.ac.uk

Abstract. Model-driven software engineering has become more popular in recent years. Due to the high number and diversity of users, new testing approaches are necessary to reduce the occurrence of faults and ensure higher quality in industrial applications. The objective of this paper is to evaluate the use of Model-Based Testing (MBT) practices in the development and execution of automated test suites to verify and validate internet-banking solutions in the context of a large banking institution in Turkey.
Keywords: Model Based Testing, Graphwalker, Internet Banking, Experience Report, Industrial Case Study

1 Introduction

Model-based techniques are also heavily used in testing phase of the software development life cycle. Model-based testing (MBT) is a type of testing that relies on explicit behaviour models that express the intended behaviours of a SUT and/or the behaviour of its environment. Test cases are generated from one of these models or their combination, and then executed on the SUT. The ideas of model-based testing, then dubbed

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

pecification-based testing, date back to the Seventies. Recent emphasis on model-based and test-centered development methodologies as well as the level of maturity of technology from the area of formal verification have led to a strong increased interest in the subject in the past decade, both in the academic field and in the industry. Model-based testing encompasses the processes and techniques for the automatic derivation of abstract test cases from abstract models, the generation of concrete tests from abstract tests, and the manual or automated execution of the resulting concrete test cases [1].

2 Industrial Contexts And Need Analysis

2.1 Industrial context and a review of the System Under Test (SUT)

Kuveyt Türk Participation Bank Inc. is a private financial (banking) institution in Turkey. The bank started its operations in the Turkish finance market in 1989. It has 414 branch locations across Turkey, and delivers a wide range of financial products and services to customers. As of year-end 2016, the bank employed about 6,000 personnel [8].

The SUT running on the bank side is the main web application used by the customers. It is called İnternet Şube (in Turkish), meaning Internet Branch, and provides user interface in Turkish and English. The app provides typical features for personal and corporate banking, e.g., transfer funds, view financial reports, and access to different bank cards owned by customers. In terms of codebase size, the app is about 293 KLOC. The app has 162 GUI screens (pages) in total, one of them is shown in Figure 1, which is the app's main window. As of this writing, the app has about 180,000 active users, and a monthly traffic of about 1.6 million logins.

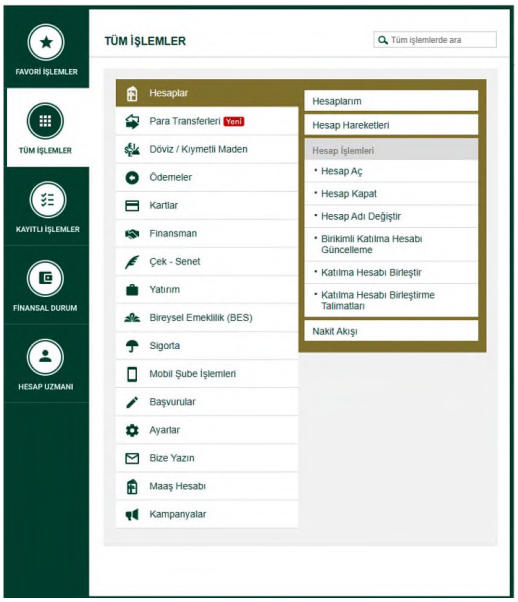


Fig. 1. Screenshot of the SUT

2.2 Need analysis and motivations fort he study

The web application was first released in 2007 for the retail and corporate customers. Furthermore, since its earliest version, the app has been undergoing all four types of software main-

tenance activities [9]: adaptive, perfective, corrective and preventive maintenance. Adaptive maintenance has been conducted since its first version to modify the system to cope with changes in the software environment (e.g., DBMS). Perfective maintenance has been conducted for implementing new or changed features. Corrective maintenance has dealt with diagnosing and fixing defects. Last but not the least, preventive maintenance has been done for increasing software maintainability, reliability, and to prevent problems in the future, e.g., using practices such as refactoring. Hence each maintenance activity on the SUT has made testing activities necessary. It goes without saying that, given the criticality of the app, its correct and uninterrupted functionality is of outmost importance. Test case generation was conducted manually, which had various drawbacks, e.g., taking too much human resources, being error prone and missing defects. Requirements are the bottom line of any application under development. MBT (Model-based testing) is very useful especially when dealing with requirements. In MBT, models are developed from the requirements for the system. The model is used to generate test cases, typically in an automated fashion. The system is run against the generated tests and the outputs are compared with the expected output. In our current SUT, requirements are written in an unstructured format and without a formal flow diagram. There was a need for strong collaboration and unambiguous develop/test base documentation between the different stakeholders: business stakeholders, system analysts, developers and software testers. Models are a good example of a visual and formal documentation. With model-based testing, requirements can be documented more systematically and showed as an overview to the stakeholders. Another advantage of model-based testing is that test case generation can be done automatically, otherwise test cases has to be gained and written manually from unstructured rich text requirements. On the ıther hand, by means of MBT, Maintenance of test cases will be decreased, because automatic test case generation can be done after changes in the requirement/model. From this perspective it is worth to mention about two main approaches for test execution in MBT; Online and Offline. Both approaches have its usages depending on the testing framework. In offline MBT, generation of test case is automated. Online MBT test case generation and execution are in motion.

3 Related Work

In Dalal, Siddhartha R., et al. [2] the report on various practices of development of tools and various methodologies related to model-based testing was generated. Some case studies have been shown which provides details and results of application of combination of test-generation methods on a large scale to diverse applications. Based on these, an insight has been given into what is practiced and what are the obstacles in transferring these technology to testing organizations. Lyu, M.R [3] offers a view of the development, testing, and evaluation schemes for software reliability, and its integration to form a unified and consistent paradigm.

Practical model-based testing: a tools approach [4] gives an insight to model-based testing in a practical manner, shows the way to write models for testing purposes and usage of model- based testing tools to generate test suites. It aims at testers and software developers who wish to use model-based testing, rather than at tool developers or academics used in specification based testing. In Fujiwara, S.et al. [5] methods for the selection of appropriate test case, an important issue line with testing of protocol implementations as well as software engineering, is shown.

Sing, Vasudha and Ramasamy, Subburaj have done a study of model-based testing. In this study firstly MBT explained, and basic MBT approach and types of model in MBT approach have been shown. Two case studies are given in their paper which will help to understand the basic concept of MBT and also describe the way to generate test suits from the model. They created their MBT model for the web application. In their

case study they have basically six pages like hotels, flights, activities, packages, cars, cruises and for each page there is individual form. Their proposed study is primarily for verifying the functionality of the SUT. And then the study showed that MBT provides the roadmap for automatic testing of software. It provides effective test strategies without missing any important test cases. Model-based testing is an efficient and adaptable method of testing software by creating a model describing the behaviour of the system under test. The availability of tools has particularly makes the job easier [6].

4 Research Design And Setup

Based on our research approach as a type of action research, after identifying the problem and needs of the project, we defined the problem as a case study that we discuss its design aspects later.

In terms of the research approach, given the nature of our project and the context, we designed an exploratory/ improving case study [10] to be executed in an observational uncontrolled context.

Stated using the GQM's goal template [20], we formulated the problem (goal) of the project as follows: the goal of the project was to develop a set of end-to-end model-based testing test suites that would cover the most important requirements of our SUT in a formalized way and generate test cases automatically from them. And finally connected the generated test cases with the test code scripts to create a chain of fully automated and ready to execute system. Based on the above goal, we posed the study's research question (RQ) as follows: What are the costs, benefits and ROI of automated testing in this industrial context? To achieve this, we conducted a case study for cost-benefit analysis of the automated test suites in the specific industrial context, from the point of view of the stakeholders involved in the project.
Given the complex nature of the project we planned a rigorous empirical approach to address the study's RQ by adopting heuristics from the literature and our recent works in this area.

5 Test Automation: Strategy, Approach, Development Of Models

In this section, we present the details of the automated testing project, including the approach, the test suites that we developed and the best practices that we applied.

5.1 An Overview of Our Test Automation Strategy

To ensure success in test automation, it is important to define and follow a proper test automation strategy [11]. Automating every case and executing every case is not possible. There is limited time and resource. We are essentially evaluating the "when to automate" question for our test automation decisions in each step, using our past experience in this area [12-14].

Automated execution is done for the regression tests, because these tests are executed frequently and on different platforms in parallel. We are also familiar with other best practice and test patterns in the area of software test-code engineering and utilized them when they were suitable for the test method under development. For example, we developed test-specific utility libraries containing utility functions which are used by multiple test methods, decoupled (separated) test data from test logic, separating the locators. In this action research we tried to automate the test case generation and execution. Some aspects of our test automation strategy are: selecting the right test cases, selecting the right test automation tool, reusability of test automation code, execution

and maintenance of test cases. Together with our stakeholders, we selected the most important/critical cases (login and money transfer cases) with the highest risks. The likelihood, impact and usage are used as a risk criteria.

5.2 Choosing the Right Test Automation Tools

An important decision in the test automation strategy is the choice of the "right" test automation tool(s) [15]. The basic idea in model-based testing is generation of models which can be transition system, UML State Machines, finite state machines, class diagrams along with constraints, etc. [7-12] from which complete test cases which is input and expected output pairs that can be generated. In online testing the model and the considered system for test are connected and the tests run dynamically whereas in offline testing the test cases/suites are generated which can be later tested with the system using a tool. A model which describes the system or software under the testing process can be taken as its partial or abstract behaviour towards the system. Hence the test cases which are taken out from a model can be said as functional tests on the similar platform of abstraction as the model. These test cases are collectively known as an abstract test suite.

Most of the available MBT tools support mainly automatic test case generation rather than test data generation and test script generation due to two reasons: firstly test case generation requires complicated strategies involving various test selection criteria from MBT models, and the generation results heavily rely on the selected criteria and strategies; secondly test case generation brings much more testing benefits, as the main efforts spent on traditional testing lies on manual preparation of test cases

There are several tools which can be used for model-based testing : Conformiq designer commercial tool in which models can be created as UML State Machines and in Qtronic Modeling Language (QML). Tests can be exported to test management tools or TTCN-3.

Spec Explorer which develops the programs in C# is a commercial tool is a successor of AsmL test tool which is now integrated with Visual Studio.

TestOptimal Tool supports FSM and E-FSM modelling with several test case generation algorithms. It has various plugins for online testing web application, windows applications, database and web services, etc. It also supports data-driven testing and pair-wise algorithms right within the model and has the facility for performing load testing using the same models.

ModelJUnit Tool allows you to write simple finite state machine (FSM) models or extended finite state machine (EFSM) models as Java classes, then generate tests from those models and measure various model coverage metrics.

TransWare BPM-X Tool creates test cases from business process models based on different criteria (statement, branch, path, condition). It can import models from several modelling tools, and can export test cases to Excel, HP Quality Center, etc.

VERA Tools is an academic tool for vulnerability testing, which allows users to define attacker models by means of extended finite state machines, and correspondently generates test cases targeting generic vulnerabilities of Web applications. In order to efficiently perform tests, VERA also provides a library containing common vulnerability test patterns for modelling [16].

MoMuT Tool is a set of model-based test case generation tools that work with UML state machine, timed automata, requirement interfaces and action systems. This tool features a fault-based test case generation strategy that allows mutations to be made on models and generates richer test cases from both original and mutated models to detect if models contain certain user-selectable or seeded faults. A fault localization mechanism is included in MoMuT for debug when a test case fails [16].

MISTA Tool is an open-source tool that generates test cases from models of finite state machine or function net. Both control-oriented and data-oriented models can be built by MISTA. The formats of test cases cover a number of languages (Java, C, C++, C#, PHP, Python, HTML and

VB) and test frameworks (xUnit, Selenium IDE and Robot framework). MISTA supports both online and offline testing [7].

GraphWalker which is an open source tool in which test cases are made from finite state machines and uses search algorithms to a-star (generating the shortest path to a specific vertex or edge) or random search algorithm to cover various states, edges, requirements. By using Grapwalker, the system model and requirements can be easily modelled in a graph based components like edges and vertices.

After our research we decided to choose GraphWalker, because it is open-source, has complete documentation with examples and is easy to learn and use. It uses directed edges and vertices to automatically create the paths that corresponds to SUT Test Cases. Another significant reason to select the GraphWalker is the Selenium integration to run the test cases automatically for our web based internet banking.

GraphWalker is generally speaking a tool for generating offline and online test sequences from Finite State Machines. In order for MBT to understand the model; there are some simple rules to follow. Every graph must have one (and only one) vertex with the label 'Start'. This shows MBT the starting point in the graph. Every vertex in a graph must have a name. Edges usually have names, but are not required to have them. The name shall follow whatever naming convention is required of the test execution engine. Labels represent the name of the vertex or edge. Keywords are written within the label of a vertex or an edge. They must exist on their own line in the label, but not the first line. In course of the project, graph editing tool namely yEd is used which is an open source tool used to design a model.

The model in the GraphWalker is built using a finite-state machine [FSM]. A FSM (Finite-State Machine) is a model with vertices (nodes) and edges (arrows) which connects the vertices with each other. There are certain set of standards which GraphWalker uses:

- A model consists of vertices and edges.
- A vertex represents a state in the system under test.
- A vertex must have a defined label.
- You can have multiple vertices with the same name.
- An edge represents the transition between two states.
- An edge can have a have a label, but it is not mandatory.
- A model must have one and only one vertex with the label Start.
- The Start vertex must have one and only one out- edge which has a mandatory label.
- The Start vertex must have no in-edges.

An edge is a transition which can be a click on a button, a timeout, a server API call etc. The label of an edge will map against a method, function or sub routine during test execution. Best practice is to start the name with 'e_'. The reason is that it is easier to recognize the function in the test execution tool, as an edge. The same name can be reused in the model. A vertex is some kind of state that is possible to verify using some kind of oracle. A vertex must always have a label. The label of a vertex is mapped against a method, function or sub routine during test execution. Best practice is to start the name with 'v_'. The reason is that it is easier to recognize the function in the test execution tool, as a vertex.

5.3 How the Test Models are Designed

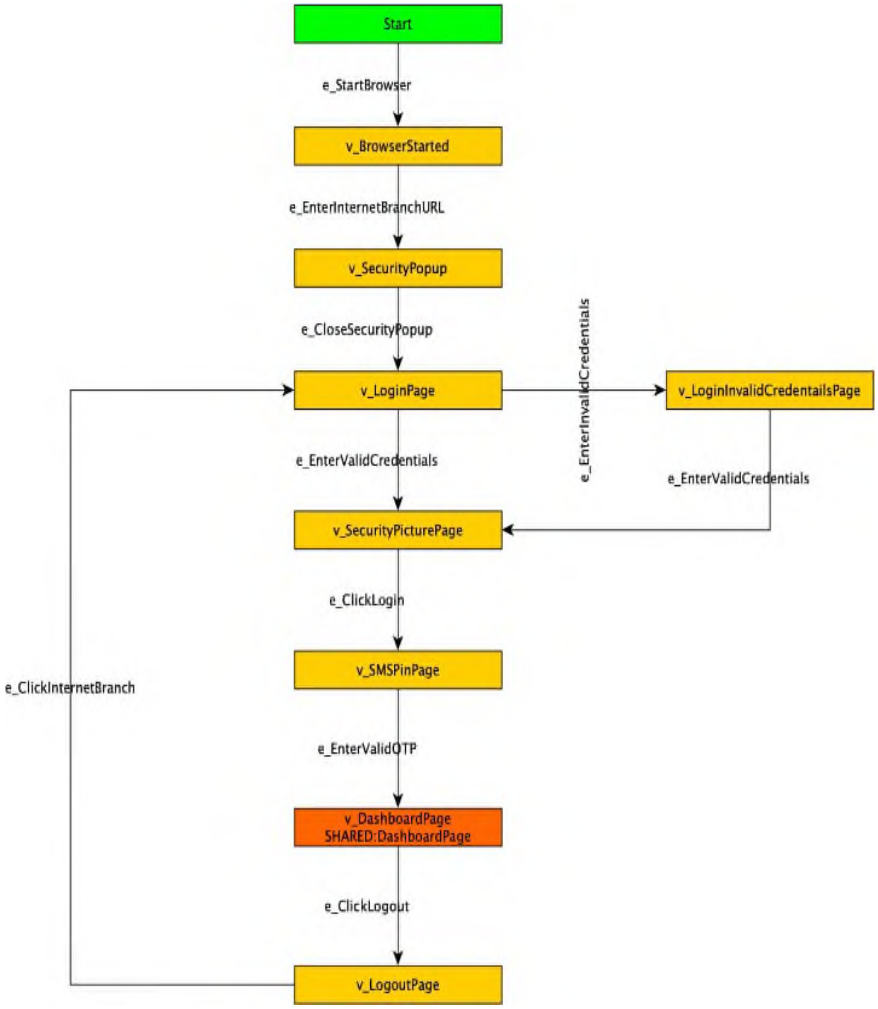


Fig. 2. Login Page Model and Test Case Generation

Our models are function oriented, simply designed without having complex models. We discussed and thought about how to model the different functionalities of our internet banking application. After these discussions we designed and reviewed the models. We tried to use small models with shared vertices to combine the models. Edges are the actions like clicks, sending input to text fields and vertices are the validations on the pages after the actions. An example is given in Figure 2; e_CloseSecurityPopup will close the popup with a click and v_LoginPage will validate whether we are on the right page. We tried to add all critical/important positive paths to the model and some important negative paths like e_EnterInvalidCredentials.

After saving the models we generated the test cases by the help of GraphWalker as paths, see Figure . Login is our base model for the whole internet banking system, because in order to

trigger a transaction in our internet banking application, any customer has to be logged in to the system. In the modelled system, since dashboard page is the main page referenced from other pages, it is labelled as SHARED. v_DashboardPage SHARED:DashboardPage is a shared vertex and combines the paths.

```
AKIN-MacBook-Pro:graphwalker akina$ java -jar graphwalker-cli-3.4.2.jar offline
{"currentElementName":"e_StartBrowser"}
{"currentElementName":"v_BrowserStarted"}
{"currentElementName":"e_EnterInternetBranchURL"}
{"currentElementName":"v_SecurityPopup"}
{"currentElementName":"e_CloseSecurityPopup"}
{"currentElementName":"v_LoginPage"}
{"currentElementName":"e_EnterValidCredentials"}
{"currentElementName":"v_SecurityPicturePage"}
{"currentElementName":"e_ClickLogin"}
{"currentElementName":"v_SMSPinPage"}
{"currentElementName":"e_EnterValidOTP"}
{"currentElementName":"v_DashboardPage"}
{"currentElementName":"e_ClickLogout"}
{"currentElementName":"v_LogoutPage"}
```

Fig. 3. Login Page Model and Test Case Generation

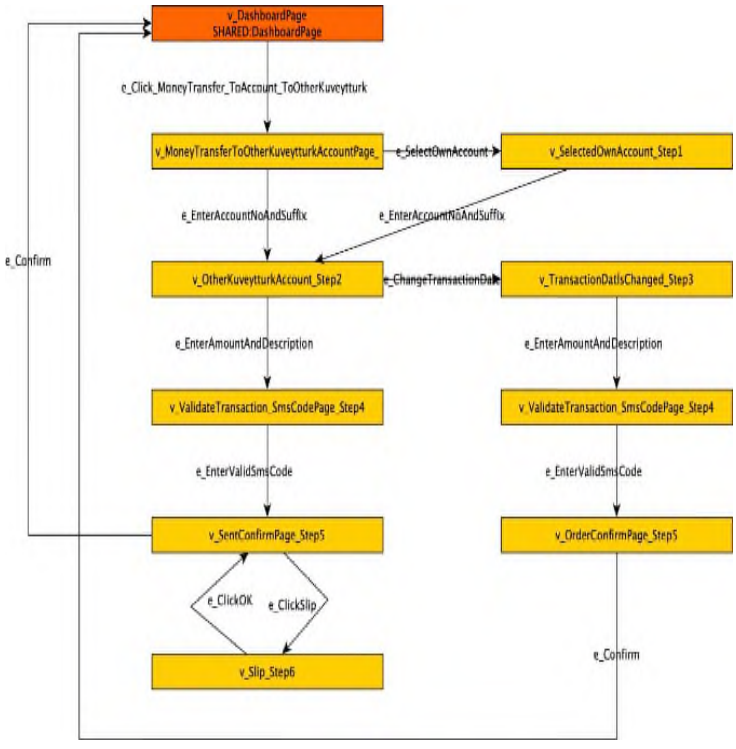


Fig. 4. Money Transfer Model

For each node and vertex we implemented test automation with Java and Selenium. This made it possible to automatically execute the generated paths. Edges starting with e_ re implemented as actions and vertexes starting with v, see Figure 4 e_EnterInvalidCredentials and v_LoginInvalid Credentials Page.

As one of the critical functions of internet banking is money transfer, money transfer models have been designed that can be seen in Figure 3. For money transfer function two main subfunctions namely money transfer to same bank account and money transfer to IBAN account use cases have been modelled.

In addition to this, Currency Exchange models, as Curency Buying and Currency Selling cases have been designed and corresponging Java Selenium Codes have been written.

```
public void e_EnterInvalidCredentials() {
    methodsPage.setObjectBy(USERNAME, customerNo);
    methodsPage.setObjectBy(PASSWORD, wrongPassword);
    methodsPage.clickObjectBy(SUBMITBTN);
}

public void v_LoginInvalidCredentailsPage() {
    String errorMessage = driver.findElement(By.cssSelector("div.innerAreaErrorShort > p")).getText();
    Assert.assertEquals(expectedErrorMessage, errorMessage);
}
```

Fig. 5. Sample Selenium Codes for Login Model

The generic actions are collected on a Library page named methodsPage to make it reusable, see Figure 5. The same is done for the different locators to increase the maintainability, see Figure 5.

```
protected WebElement setObjectBy(By by, String value) {
    WebElement element = getElementBy(by);
    element.clear();
    element.sendKeys(value);
    return element;
}

protected WebElement clickObjectBy(By by) {
    WebElement element = driver.findElement(by);
    waiter.until(ExpectedConditions.presenceOfElementLocated(by));
    element.click();
    return element;
}

public WebElement getElementBy(By by) {
    return driver.findElement(by);
}
```

Fig. 6. Reusable methods and variables declaration

```
By USERNAME = By.id("IntUserName");
By PASSWORD = By.id("Password");
By SUBMITBTN = By.id("btnSubmit");
By LOGIN = By.id("LoginButton");
```

Fig. 7. Reusable methods and variables declaration

6 Evaluation of The Approach

In terms of quantitative benefits the following improvements were noticed: time to create the test basis, test base maintenance, development time, time to generate the test cases, test case maintenance.

As also for quantitative benefits, Time to maintain the test basis, time to generate and maintain the test cases is decreased.

Shown in Table 2 time to create the test basis did not change but other benefits are better on the model based testing side. Time to generate the test cases decreased with %50 compared to manual test case writing. As shown as Table 2 time is 2 days for to generate the test cases before model based and then time is 1 day after model based. Especially test case maintenance is reduced, because test cases can automatically generated. Test case maintenance time fallen from 1 day to 2 hours as shown in the Table 2. When there is a change applied in the test basis (changed requirement) test cases has to be maintained. In the traditional way a software test engineer has to read the changes and maintain the test cases, but with MBT test cases can be generated automatically. This means that test case maintenance is unnecessary. So testing can be done faster and more reliable. This provides to decrease human effort to create the test cases.

In terms of qualitative and intangible benefits the following improvements were noticed: testability, close collaboration among stakeholders (business stakeholders, analysts, developers, testers), ambiguity of requirements, human effort to create and maintain the test cases. A test basis of tens of pages can be shows in one model which has a clear overview of the functionality. This provides also easy maintenance of the test basis, developers detect changed requirement so development time decreases and automated test's testability increases.

Shown in Table 2 all qualitative and intangible benefits are changed positively. In general MBT approach provided faster software development life cycle with less effort. All evaluation are summarized in the Table 2 that it shows as numeric values and ratings.

7 Discussions

In our test models like login and money transfer, we tried to keep our models simple and easy to understand. From this perspective we made our system models modular and reusable. We connected the modular models together with SHARED attribute and tried to avoid developing big complex models. Our internet banking web application has about 178 functionalities and modelling it in one model would make it complex and unreadable. Giving the models to GraphWalker we automatically generated the test cases. These test cases can be executed manually, but we automated the execution with Java-Selenium.

Each vertex in our model is a validation and we tried to validate only the most critical points on the page. Developing a strategy is important for model based testing, otherwise adding each detail in one model will make the model complex and unreadable. This is also the case for writing the automation code, validating each component in a page can fail an execution because of insignificant details. Too detailed models will result in more test cases, more test cases will lead to more test automation code and these will all cause the increase in execution time.

In our test models, we have firstly generated the test cases with the random 100% edge coverage and GraphWalker generated some redundant steps. Redundant test cases cause an increase in the number of test cases and test execution time. By taking advantage of quick-random generator feature in GraphWalker we could have chance to result in less redundant steps.

8 Conclusisons And Future Work

Models are very important in model based testing. They have to be complete and well organized. Automated test case generation is the most important benefit of model based testing and also we can see the whole picture of the SUT shown in the model. On the other hand, collaboration with the stakeholders about the SUT has improved in terms of the clear overview in the model.

For generating the test cases we used random generator algorithms. As future work, trying different path generation algorithms which are supported by GraphWalker can be quite challenging and the results can be analysed within each other. As a result of comparison of different path generation algorithms, the number of generated test cases and test execution time can be decreased.

GraphWalker will release "shortest_all_paths" generator algorithm and this algorithm will calculate and generate the shortest path through the model. This generator is not available yet, so we could not try it. As a future work when this feature is available, it could be tried for further detailed analysis of the paths of the graph.

Another interesting future work can be the usage of weights and guards in our models. Using these futures we can guide the generation of test case steps. Weights are used only when using the random generator. It holds a real value between 0 and 1, and represents the probability that the edge should be chosen. Guards are a mechanism only associated with edges. Their roles are the same as an if-statement, and makes an edge eligible or not for being walked.

Table 1. Summarizing the benefits of model based testing

Types of benefits	**Aspects**	**Regular manual text based vs model based**	
		Before model based	**After model based**
Quantitative	Time to create the test basis	4 days	4 days
	Test base maintenance	2 days	1 day
	Development time	8 days	6,5 days
	Time to generate the test cases	2 days	1 day
	Test case maintenance	1 day	2 hours
Qualitative and intangible	Testability	Low	High (Automated test case generation)
	Close collaboration among business stakeholders, analysts, developers and testers	Low	High (Visible, clear models)
	Ambiguity of requirements	Low	High (Visible, clear models)
	Human effort to create the test cases	High	Low (Automated test case generation)

Acknowledgnment

This work was in part supported by the European ITEA3 TESTOMAT (the Next Level of Test Automation) project (www.testomatproject.eu) and the Scientific and Technological Research Council of Turkey (TÜBİTAK) via grant #9180076.

References

1. UTTING, Mark; PRETSCHNER, Alexander; LEGEARD, Bruno. A taxonomy of model-based testing approaches. Software Testing, Verification and Reliability, 2012, 22.5: 297-312.

2. Dalal, Siddhartha R., et al. "Model-based testing in practice." Proceedings of the 21st international conference on Software engineering. ACM, 1999
3. Lyu, M.R. 1998. An integrated approach to achieving high software reliability. Aerospace Conference, 1998. Proceedings. , IEEE (1998), 123 – 136
4. Utting, Mark, and Bruno Legeard. Practical model-based testing: a tools approach. Morgan Kaufmann, 2010
5. Fujiwara, S., Bochmann, G., Khendek, F., Amalou, M., and Ghedamsi, A., "Test Selection Basedon Finite State Models", IEEE Transactions on Software Engineering, Vol. 17, No. 6., June 1991
6. International Journal of Scientific & Engineering Research, Volume 6, Issue 2, February-2015 941 ISSN 2229-5518 pp.934-941
7. Li Wenbin, Gall Franck Le, Spaseski Naum, A Survey on Model- Based Testing Tools for Test Case Generation, The 4th International Conference on Tools and Methods of Program Analysis, At Moscow, Russia, Volume: 779
8. Kuveyt Türk Participation Bank Inc., "2018 annual report," https://www.kuveytturk.com.tr/medium/document-file- 2317.vsf Last accessed: June 2019.
9. D. J. Reifer, Software Maintenance Success Recipes. CRC Press, 2012.
10. P. Runeson, M. Host, A. Rainer, and B. Regnell, Case Study Research in Software Engineering: Guidelines and Examples. John Wiley & Sons, 2012.
11. D. Graham and M. Fewster, Experiences of Test Automation: Case Studies of Software Test Automation. Addison-Wesley Professional; 1 edition, 2012.
12. V. Garousi and M. V. Mäntylä, "When and what to automate in software testing? A multi-vocal literature review," Information and Software Technology, vol. 76, pp. 92-117, 2016.
13. Z. Sahaf, V. Garousi, D. Pfahl, R. Irving, and Y. Amannejad, "When to Automate Software Testing? Decision Support based on System Dynamics – An Industrial Case Study," in Proc. of International Conference on Software and Systems Process, 2014, pp. 149-158.
14. V. Garousi and E. Yıldırım, "Introducing automated GUI testing and observing its benefits: an industrial case study in the context of law- practice management software," in Proceedings of IEEE Workshop on NEXt level of Test Automation (NEXTA), 2018, pp. 138-145.
15. P. Raulamo, M. V. Mäntylä, and V. Garousi, "Choosing the right test automation tool: a grey literature review," in International Conference on Evaluation and Assessment in Software Engineering, Karlskrona, Sweden, 2017, pp. 21-30.
16. LI, Wenbin; LE GALL, Franck; SPASESKI, Naum. A survey on model-based testing tools for test case generation. In: International Conference on Tools and Methods for Program Analysis. Springer, Cham, 2017. p. 77-89.

Aygıt Ağacı Yazılımlarının Model Güdümlü Geliştirilmesinin Tersine Mühendislik ile Desteklenmesi

Sadık Arslan[1,2], Geylani Kardaş[2]

[1] Kent Kart Ar-Ge Merkezi, Kent Kart Ege Elektronik A.Ş., İzmir, Türkiye
[2] Ege Üniversitesi, Uluslararası Bilgisayar Enstitüsü, İzmir, Türkiye
sadik.arslan@kentkart.com.tr, geylani.kardas@ege.edu.tr

Özet. Aygıt Ağacı (DT) yapıları bir gömülü sistem donanımının içerisindeki fiziksel cihaz bileşenlerinin düğümler ile tanımlanmasını sağlamaktadır. DT kaynak dosyalarının metin-tabanlı ve güncel programlama dillerinden farklı bir yapıdaki sözdizimlerini kullanmadaki zorluğu aşmak ve çeşitli işlemci mimarileri için DT kodlarını otomatik üretmek için model güdümlü geliştirme kullanılabilir. Ancak bazı durumlarda üretilen DT'ler üzerinde yeni donanım ekleme, donanım özelliği değiştirme, vb. amaçlarla kod ekleme veya değiştirme gerekmektedir. Bu bildiride DSML4DT adı verilen bir alana-özgü modelleme dili kullanılarak otomatik oluşturulan DT kodlarında sonradan yapılan değişikliklerin modellere yansıtılmasını ve daha önce başka yöntemlerle oluşturulmuş DT kodlarından yazılım modellerini elde etmeyi sağlayan bir tersine mühendislik yöntemi tanıtılmaktadır. Önerilen tersine mühendislik yaklaşımının toplu taşıma bilgi sistemleri üreten bir firmanın donanımlarında ihtiyaç duyulan DT tabanlı yazılımların geliştirilmesinde uygulanması örnekleri verilmiş ve uygulama sonuçları değerlendirilmiştir.

Anahtar Kelimeler: Model-güdümlü Geliştirme, Alana-özgü modelleme dili, Aygıt Ağacı, Gömülü Yazılım, Tersine Mühendislik.

Reverse Engineering Support for the Model-driven Development of Device Tree Software

Sadik Arslan[1,2], Geylani Kardas[2]

[1] Kent Kart R&D Center, Kent Kart Ege Elektronik A.S., Izmir, Turkey
[2] Ege University, International Computer Institute, Izmir, Turkey
sadik.arslan@kentkart.com.tr, geylani.kardas@ege.edu.tr

Abstract. Device Trees (DTs) provide the description of physical devices inside an embedded system hardware with node specifications. Model-driven development may be used for dealing with the difficulties encountered in DT de-

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

velopment due to the text-based syntax of DT source files which has a different structure from the well-known general purpose programming languages; and hence also enables automatic DT code generation for various processor architectures. However, in some situations, there needs later modifications on the generated DT codes for adding new devices, changing device specifications, etc. In this paper, we introduce a reverse engineering method which supports both 1) reflecting the changes made in DT codes auto-generated by using a domain-specific modeling language, called DSML4DT, into models and 2) generating models from the existing DT codes produced with any other methods. The application of the proposed approach is exemplified and evaluated with the implementation of DT based software for the various hardware manufactured by a company whose main expertise is on the public transportation systems.

Keywords: Model-driven Development, Domain-specific modeling language, Device Tree, Embedded Software, Reverse Engineering.

1 Giriş

Donanım bilgisi ve yapılandırmasını içeren Aygıt Ağacı (DT) dosyaları [1] çeşitli gömülü platformlar için işletim sistemlerinin derlenmesi sırasında kullanılırlar. DT'ler bir gömülü sistem donanımının içerisindeki fiziksel cihaz bileşenlerinin düğümler ile tanımlanmasını sağlamaktadır [2]. Ancak gömülü sistem yazılımı geliştiricileri DT kaynak dosyalarının metin-tabanlı ve güncel programlama dillerinden farklı bir yapıdaki sözdizimlerini kullanmada çoğu zaman zorlanmaktadır. Yeni her gömülü platform için ihtiyaç duyulan DT dosyaları baştan sona ve ayrı ayrı hazırlanmalıdır. Bununla birlikte geliştiricilerin DT dosyalarını hazırlaması için mikro-işlemciye özel donanımlara da hakim olması gerekmektedir. İlgili donanımlar hakkında alan bilgisine sahip ancak yazılım geliştirme bilgi ve deneyimi az olan mühendisler için bu dosyaların hazırlanması aşaması olduça zahmetlidir. Üstelik farklı mikro-işlemci mimarileri ve çeşitleri için DT yazılımlarına ait bileşenlerin kodlanması ve konfigüre edilmesi yine birçok geliştirici için zor ve zaman alıcı olmaktadır. Literatürde DT tabanlı gömülü sistem yazılımı geliştirmeye dair çeşitli çalışmalar [3-9] bulunmakla birlikte bu çalışmalar daha çok DT konfigürasyonu oluşturmanın örneklerini sunmakta ve yukarıda sözü edilen DT yazılımı geliştirme zorluklarının giderilmesini göz önüne almamaktadır.

Model güdümlü geliştirme teknikleri DT yazılımlarına ait kaynak dosyalarının farklı işlemci mimarileri ve çeşitleri için otomatik üretilmesini sağlayabilir ve yine yukarıda listelenen yazılım geliştirme süreci zorluklarını azaltabilir. Bu amaçla daha önceki çalışmamızda [10] DT yazılımlarının modellenmesi için kullanılabilecek bir üstmodel ve somut bir sözdizim geliştirilmiştir. Bu sözdizimi kullanan ve bir dönüşümsel semantik ile DT modellerinden gömülü sistemlerde ihtiyaç duyulan DT yazılımlarının otomatik elde edilebildiği DSML4DT adı verilen bir alana-özgü modelleme dili (DSML) oluşturulmuştur.

Her ne kadar DSML4DT ile DT yapılarının tüm bakışaçılarına hitap eden modeller oluşturulabiliyor ve bu modellerden kodlar üretilebiliyor olsa da DSML4DT'nin endüstriyel boyutta kullanımları sırasında bazı gömülü sistem donanımları için

oluşturulan bu otomatik kodlara yeni eklemelerin ya da değişikliklerin yapılmasının gerektiği durumların da oluştuğu gözlenmiştir. Bazı durumlarda da DT kodunun önceden bir yazılım modeli yoktur ancak bu kodların DSML4DT ile yazılım geliştirme sürecine entegre edilmesi için modellere ihtiyaç duyulabilmektedir.

Yazılımların tersine mühendisliği yazılımların yönetilmesine ve zaman içerisindeki değişimlere göre evrimleşmesine yardımcı olmaktadır [11]. Kaynak kodlar incelenerek yazılım sistemlerinin ana elemanları belirlenebilir ve yazılımın önceden tanımlanmış bir düzeydeki soyutlaması elde edilebilir [12]. Bu bildiride, hem DSML4DT'nin kod üretimi mekanizması sonucunda oluşturulmuş DT kodlarındaki değişikliklerin DSML4DT modelerine yansıtılmasını hem de daha önce başka yöntemlerle oluşturulmuş DT kodlarından yazılım modellerini elde etmeyi sağlayan bir tersine mühendislik yöntemi tanıtılmaktadır.

Bildirinin 2. bölümünde DT yapıları ve DSML4DT dili hakkında kısaca bilgi verilmiştir. DT yazılımlarının geliştirilmesi için önerilen tersine mühendislik yöntemi 3. bölümde anlatılmıştır. Tersine mühendislik yaklaşımının ve geliştirilen aracın kullanılmasını örnekleyen bir durum çalışması 4. bölümde yer almaktadır. Bölüm 5'te, ilgili literatürdeki önceki çalışmalar anlatılmış ve önerilen yöntemin farkları belirtilmiştir. Son bölümde çalışmanın bir değerlendirmesi, elde edilen sonuçlar ve ileriye yönelik çalışma hedefleri bulunmaktadır.

2 DT Yapıları ve DSML4DT

Bir DT, donanım tanımı sağlamaktadır ve her aygıtın düğümler şeklinde ve özellikleri ile birlikte temsil edildiği bir ağaç yapısına sahiptir. DT, çalışılan gömülü sistemin işlemcisi, belleği, veri yolları ve çevre birimleri gibi birçok bölüm hakkında bilgi içerir. İşletim sistemi, önyükleme (ing. bootloading) sırasında DT yapısını ayrıştırır ve mikroişlemciyi nasıl yapılandıracağını burada belirler. DT yapısı, "/" karakteri ile gösterilen ve kök olarak adlandırılan bir düğümle başlamaktadır. Ağaç yapısına benzer bir şekilde, DT'de ebeveyn konumundaki her düğümden çok sayıda buluan, ismi ve numarası olan birden fazla çocuk düğüm oluşturulabilmektedir. Düğümler isteğe göre, ek veri içeren nitelik değerlerine sahip olabilmektedir. Metinsel hazırlanan DT tanımları, ses kartı, ekran kartı gibi birçok farklı arayüz ve özelliğin ayarlanması için elle hazırlanması gereken çok sayıda koddan oluşmaktadır [6].

DSML4DT, uluslararası DT standardına [1] uygun olarak gömülü sistem DT yazılımlarının model-güdümlü geliştirilmesini sağlamaktadır. DSML4DT üstmodeli toplamda 70'ten fazla DT bileşeni ve bunların ilişkilerini içermektedir. Üstmodel farklı DT modelleme bakış açılarına göre modellemeyi sağlayan 5 alt üstmodelin birleşiminden oluşmaktadır. Bunlar: Temel işlemci ve bellek birimlerinin tanımlandığı *Core*, yonga üzeri sistem entegre devrelerinin özelliklerinin modellenebildiği *SoC*, düşük bant genişliğine sahip SoC çevresel bileşenlerinin hazırlanabildiği *Aips_Bus*, harici birimlerlerle iletişimi paylaşımlı çevresel hat arayüzü üzerinden kurgulayan *Spba_Bus* ve yonga üzerinde bulunmayıp yongaya dışardan bağlı çevresel birimlerin modellenebilidği *Peripheral*.

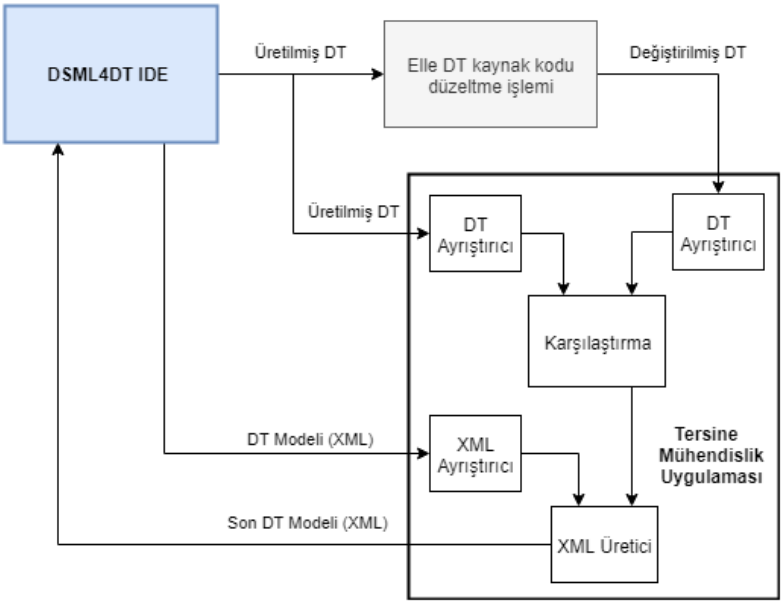
Yukarıda isimleri verilen DT bakışaçılarının her birindeki üst-varlıklar için DSML4DT'de görsel somıt sözdizim notasyonları vardır ve her bir bakışaçısı için ayrı ayrı sistem modelleri görsel bir şekilde oluşturulabilmektedir. DSML4DT'nin entegre geliştirme ortamı (IDE) çeşitli model-güdümlü araçların geliştirilmesinde yaygın olarak kullanılan Eclipse tabanlı Sirius platformu üzerine kurgulanmıştır. DSML4DT IDE'sinde yer alan bir palette her bakış açısı için görsel notasyonlar listelenmektedir. Yazılım geliştiriciler ilgili paletten sürükle-bırak yöntemi ile kendi DT modelleri için gerekli elemanları IDE'deki modelleme ortamına getirirler. Modelleme sırasında DSML4DT'nin statik semantik kuralları ve diğer model kısıtları doğrudan işletilir ve yanlış bir model oluşturma durumunda geliştirici uyarılır. Model doğrulama tamamlanmadan modeller XMI ile serileştirilerek saklanamaz ve kod üretimi yapılmaz. Oluşturulan görsel DT modellerinden otomatik DT kodlarının üretimi Sirius'a entegre Acceleo'da yazılan modelden metne kuralların işletilmesi ile sağlanmaktadır. Her bir DSML4DT model örneği üzerinde Accelo dili kullanılarak yazılan dönüşüm kuralları işletilir ve sisteme ait DT kodları .dts dosya formatında oluşturulur. Yer kısıtları nedeniyle bu bildiride gösterilemeyen DT yapısı örneğine, DSML4DT'nin sözdizimine ve IDE'nin görünümüne [10]'dan erişilebilinir.

3 DSML4DT için Tersine Mühendislik Yöntemi

Mevcut DT yazılımlarından DSML4DT'ye uyan DT modellerinin otomatik oluşturulması için bir yöntem bu çalışmada önerilmektedir. Geliştirilen tersine mühendislik yaklaşımının hayata geçirilebilmesi için de Java dilinde yazılmış bir uygulama geliştirilmiştir. Bu uygulama temelde bir ayrıştırma (ing. parse) işlemi yapmaktadır. DSML4DT IDE'si kullanılarak hazırlanan her bir görsel DT modeli DSML4DT semantik kurallarına göre geçerliliği kontrol edildikten sonra Genişletilebilir İşaret Dili (XML) formatındaki bir dosya içerisinde saklanmaktadır. DT yazılımları için tersine mühendisliği sağlayan bu uygulama DT kod dosyalarını ve XML formatında saklanan DT model dosyalarını girdi olarak almakta ve çıktı olarak yine DT model dosyaları üretmektedir. Ayrıştırma uygulaması hem DSML4DT ile önceden oluşturulmuş kodlar (senaryo 1) hem de herhangi bir DT modeli bulunmayan ve DSML4DT ile geliştirilmemiş kodlar üzerinde (senaryo 2) çalışarak DT yazılım modellerini otomatik oluşturabilmektedir. Uygulama bu bildiri hazırlandığı tarihte DSML4DT IDE'sine entegre değildir ve IDE'den bağımsız olarak çalışmaktadır.

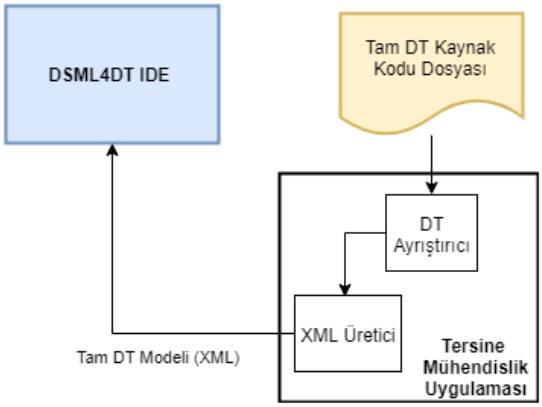
Uygulama, DSML4DT tarafından üretilmiş olan yazılım kodlarını içeren dosyaları aldığında öncelikle bunlar üzerinde bir ayrıştırma yapmaktadır. Ağaç yapısındaki DT dosyalarında düğümlerin bulunması gerekmektedir. Ebeveyn ve çocuk düğümlerin ilişkisi DT yapısının temelini oluşturmaktadır. Bu düğümler DT'nin sözdiziminde "{" ve "}" karakterleri arasında bulunmaktadır. Çocuk düğümler de üst ebeveyn düğümün içinde yine bu karakterler arasına yerleşmektedir. Uygulama temelde dosyalardaki karakterleri tek tek taramakta ve bu karakterlere göre düğümleri oluşturmaktadır. Ağaç yapısı, bulunan düğüm sınırlarına ve ayrıştırılan düğüm isimlerine göre oluşturulmaktadır. DT kaynak kodu üzerinde yapılan bu ayrıştırmadan sonra DSML4DT modeline ait XML formatındaki dosya üretilmektedir.

Senaryo 1'de DSML4DT ile modellenen ve bu modelden üretilen DT yazılım kodlarında daha sonra yazılım geliştiricinin manuel yaptığı değişikliklerin (yeni DT elemanları için kod ekleme, mevcut DT yapısına ait kodu günceleme, vb.) yazılım modeline yansıtılması göz önüne alınmaktadır (Şekil 1). Mevcut DSML4DT örnek modelinden otomatik elde edilen (değiştirilmemiş) DT kodları ve manuel değiştirilmiş DT kodları yukarıda tarif edilen uygulama ile ayrıştırılır. Ayrıca uygulama mevcut görsel modelin XML formatında saklanan versiyonu üzerinde de benzer bir ayrıştırma uygular. Ayrıştırılan kaynak kodu dosyaları (otomatik ve manuel değiştirilen) yine bu uygulama ile karşılaştırılır ve ağaç yapıları belirlenir. İki dosya arasındaki farka göre de mevcut DSML4DT örnek modeli XML dosyasındaki değişiklikleri yaparak güncellenmiş DT modelini oluşturmaktadır. Bu yeni modele ait XML dosyası DSML4DT IDE'sinde açıldığında yazılımcının manuel yaptığı değişikliklere göre güncellenmiş DT modelinin grafiksel (görsel) hali görüntülenebilmektedir.



Şekil 1. DT yazılımları için geliştirilen tersine mühendislik uygulaması – Senaryo 1

Bu çalışmada geliştirilen tersine mühendislik yaklaşımının uygulanabileceği 2. senaryo ise DT yazılım modeli halihazırda olmayan ve/veya DSML4DT dili kullanılmadan oluşturulan DT kaynak dosyalarına karşılık gelen yazılım modellerinin otomatik oluşturulmasıdır. Herhangi bir DT kaynak kodu yine uygulamanın ayrıştırıcı kısmından geçirilerek bu dosyada tanımlı tüm DT yapılarının ve özelliklerinin görsel gösterimini sağlayacak model dosyasının XML ile "encode" edilmiş hali otomatik oluşturulmaktadır. Bu dosya DSML4DT IDE'sinde açıldığında model yine dilin görsel notasyonu ile yazılım geliştiricilere gösterilir. Böylece başka araçlarla ya da yöntemlerle oluşturulmuş DT dosyaları da DSML4DT ortamına dahil edilebilir.



Şekil 2. DT yazılımları için geliştirilen tersine mühendislik uygulaması – Senaryo 2

Uygulamanın çalışmasındaki temel Java programı *DTS_RevEn* olarak adlandırılmıştır. Bu programda öncelikle dışarıdan girilen otomatik üretilmiş ve elle değiştirilmiş DT kaynak dosyaları alınmaktadır. Bu programda kullanılan *dtsParse* objeleri ayrıştırma işlemlerini yapmaktadır. Programa alınan DT kaynak kodu dosyaları içerisinde bir döngü aracılığı ile satır satır gezilerek düğümler belirlenir ve nesne koleksiyonlarına dahil edilir. Bu koleksiyon nesnelerinde, girdi konumundaki otomatik üretilmiş DT kodları ve üzerinde değişiklik yapılan (manuel) kodlar arasında karşılaştırmalar ve diğer işlemler kolaylıkla yapılabilmektedir. *listGen* ve *listMod* isimli ve Java *Collection* tipinde oluşturulmuş dizilerde DT dosyalarının düğümlerinin isimleri tutularak işlemler yapılmaktadır. Programın *dtsParse* sınıfından türetilen *dtsGen* ve *dtsMod* nesneleri ile *phraseGen* ve *phraseMod* karakter değişkenleri kullanılarak ayrıştırma yapılmaktadır. *findParanthesis* metodu ile DT kaynak dosyasındaki düğüm parantezleri bulunmaktadır. Daha sonra kullanılan *findNodes* metodu ile DT kaynak kodlarındaki düğümler bulunmaktadır. *generatedList* ve *modifiedList* dizi listeleri üzerinden de elle eklenen DT kodlarının farkı *removeAll* metodu ile belirlenmektedir. Bu işlemlerden sonra *generateXML* metotu ile DSML4DT IDE'sinde açılabilecek DT modeli XML dosyası üretilmiş olmaktadır. Şekil 3'te uygulamanın yukarıda anlatılan temel adımlarını tesmil eden basit sözde kod (ing. pseudocode) verilmiştir. Senaryo 1 için Şekil 3'te listelenen tüm adımlar uygulanmaktadır. Senaryo 2 için sadece satır 2, satır 8-10 ve satır 13'ün işletilmesi yeterlidir. *generateXML* metodu ayrıştırılmış olan DT kaynak kodlarından farka göre yeni DT modelinin XML'ini oluşturmaktadır. Bu amaçla DOM XML ayrıştırıcı ve üreticiyi kullanır. Burada yapılan işlemlerde mevcut DT modeli ayrıştırılmakta ve ilgili yerlere yeni XML satırları eklenmektedir. Her XML satırı bir DT düğümüne denk gelmektedir. Tüm DT dosyasının oluşturulması benzer mantıkla yapılmaktadır. DT'nin ağaç yapısı üzerinde gezilirken yeni XML dosyası da oluşturulmaktadır. Oluşturulan XML dosyası DSML4DT içerisinde açılabilir ve işletilebilir bir haldedir.

```
01  Adding generated DTS file
02  Adding modified DTS file
03
04  Parsing generated DTS()
05      Finding Paranthesis(generated DTS file)
06      Finding Nodes(generated DTS file)
07
08  Parsing modified DTS(modified DTS file)
09      Finding Paranthesis(modified DTS file)
10      Finding Nodes(modified DTS file)
11
12  Printing different nodes(delta nodes)
13  Generating XML file
```

Şekil 3. DT tersine mühendislik uygulaması sözde kodu

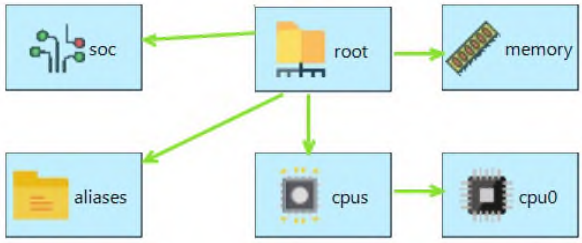
4 Durum Çalışması

Bu bildiride önerilen tersine mühendislik yaklaşımı toplu taşımada kullanılan bir araç sürücü terminaline ait gömülü sistemin DT tabanlı geliştirilmesi için uygulanmıştır. Uygulama T.C. Sanayi ve Teknoloji Bakanlığı'nca 2013 yılından beri akredite olan Kent Kart Ege Elektronik A. Ş. (kısaca Kentkart) firmasının Ar-Ge merkezinde gerçekleştirilmiştir. Kentkart otomatik ücret toplama, araç takibi ve yönetimi ve gerçek zamanlı yolcu bilgilendirme gibi çeşitli alanlarda toplu taşıma bilgi sistemleri üreten bir firmadır. Kentkart aynı zamanda toplu taşımada kullanılan ücret toplama makineleri, turnike validatörleri ve araç sürücüsü terminalleri gibi donanımları da üretmektedir. Tüm bu donanımlar üzerinde çalıştırılan yazılımların önemli bir kısmı DT tabanlıdır ya da DT yapılarına göre konfigüre edilmektedir.

Çalışmada kullanılan araç sürücüsü terminali Kentkart firması tarafından üretilen ve çift çekirdekli iMX6 işlemcisine sahip bir gömülü sistemdir. Bu tip bir cihazın DT yapısı tasarlanırken toplamda 100 civarında farklı özelliği içeren DT düğümünün hazırlanmasına ihtiyaç vardır.

Bu örnek çalışmada söz konusu sürücü terminalinin DSML4DT kullanılarak oluşturulan Core bakış açısı modelinden elde edilen otomatik kodlarda Kentkart mühendislerinin manuel gerçekleştirdiği bir değişikliğin bu bildirinin 3. bölümünde anlatılan yöntem kullanılarak sistemin DT modeline nasıl yansıtıldığı göz önüne alınacaktır. DSML4DT Core bakış açısında bir gömülü sistemdeki işlemci ve bellek gibi temel blokları tanımlayan düğümler ve bunların birbirleri ile ilişkileri modellenir. Burada bulunan *root* düğümü tüm DT yapısının türetildiği düğümdür ve bu düğümden işlemciler için *cpus*, ana bellek için de *memory* düğümleri türetilmektedir. Sistemin geri kalan tüm özellikleri de bu bakış açısındaki *soc* düğümünde bulunmaktadır. Ancak bu bakış açısında *soc* düğümüne bağlı olan çocuk düğümler görülememektedir. Bu düğümler farklı bir bakış açısı olan Soc bakış açısında bulunmaktadır. DSML4DT'nin bu bakışaçıları için detaylı bilgi [10]'da bulunmaktadır.

DSML4DT dili kullanılarak araç sürücüsü terminali için oluşturulan örnek Core modelinin ilk hali Şekil 4'te verilmiştir. Sistemde işlemci çekirdekleri *cpus* ebeveyn düğümünden türetilmektedir. Bu DT modelinde şu an sadece *cpu0* adında bir çekirdek olduğu görülebilmektedir.



Şekil 4. Araç sürücü terminali DT yapısı Core modelinin ilk hali

Şekil 4'teki DT modelinden otomatik kod üretimi yapıldığında DSML4DT IDE aracılığıyla Şekil 5'te satır 1-26 arasında görülen DT kaynak kodları elde edilmektedir. Buradaki kaynak kodunda "/" karakteri ile başlayan yerde *root* düğümü bulunmaktadır. *root* düğümünün *model* ve *compatible* isminde öznitelikleri vardır. Ardından gelen *cpus* düğümü *root* düğümünün çocuğudur. *cpu0* düğümü de *cpus* düğümünün çocuğu durumundadır. Otomatik üretilen bu koda yazılım geliştirici daha sonra ikinci bir cpu tanımını manuel eklemiştir. Şekil 5'te koyu olarak verilen (satır 27-32 arasındaki) kodlar *cpu1* adlı bu düğüm için manuel eklenen kodlardır.

```
01  /{
02      model = Kentkart i.MX6 DualLite Smart Device Board;
03      compatible = "fsl,imx6dl-sabresd", "fsl,imx6dl";
04
05      cpus{
06          address-cells = <1>;
07          size-cells = <0>;
08
09          cpu0{
10              compatible = "arm,cortex-a9";
11              device_type = "cpu";
12              reg = <0>;
13              next-level-cache = <&L2>;
14              operating-points = <
15                  996000 1275000
16                  792000 1175000
17                  396000 1150000
18                  >;
19              fsl,soc-operating-points = <
20                  996000 1175000
21                  792000 1175000
22                  396000 1175000
23                  >;
24              clock-latency = <61036>;
25              };
26
27          cpu1{
28              compatible = "arm,cortex-a9";
29              device_type = "cpu";
30              reg = <1>;
31              next-level-cache = <&L2>;
32              };
33      };
```

Şekil 5. Otomatik üretilmiş ve elle eklenmiş DT kod örneği

```
01  <devicetree_viewpoint:devicetree>
02    <root compatible="&quot;fsl,imx6dl-sabresd&quot;, &quot;fsl,imx6dl&quot;"
03        model="Kentkart i.MX6 DualLite Smart Device Board" name="root">
04      <cpus address_cells="1" name="cpus" size_cells="0">
05          <cpu0 clock_latency="61036" clock_names="&quot;arm&quot;,
06              &quot;pll2_pfd2_396m&quot;, &quot;step&quot;,&quot;pll1_sw&quot;,
07              &quot;pll1_sys&quot;, &quot;pll1_bypass&quot;, &quot;pll1&quot;,
08              &quot;pll1_bypass_src&quot;" compatible="&quot;arm,cortex-a9&quot;"
09              device_type="&quot;cpu&quot;" name="cpu0"
10              next_level_cache="&amp;L2"
11              operating_points="996000 1275000 792000 1175000 396000 1150000"
12              reg="0" soc_operating_points="996000 1175000 792000 1175000 396000
13                  1175000"/>
14      </cpus>
```

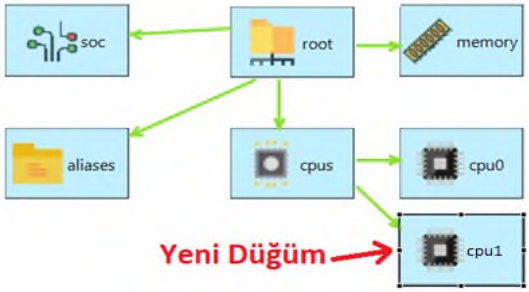
Şekil 6. DSML4DT ile hazırlanan görsel DT modeline ait XML dosyasından bir kısım

Şekil 4'teki DT modelinin DSML4DT IDE'si içerisindeki dosyasında görsel modeldeki bu düğümlerin ve özniteliklerin XML ile serileştirilmiş metinsel halleri tutulmaktadır. Şekil 6'da bu dosyadan *root*, *cpus* ve *cpu0* düğümlerinin yer aldığı bir kısım verilmiştir.

Buradaki uygulama bildirinin 3. bölümünde anlatılan tersine mühendislik yaklaşımının kullanım senaryolarından birincisinin bir örneğidir. Şekil 5'te görülen ve manuel eklemelerin yapıldığı koddaki değişikliğin Şekil 6'daki DT Core modeli XML dosyasına yansıtılabilmesi için geliştirdiğimiz tersine mühendislik uygulaması eklemelerin yapıldığı kod üzerinde çalışarak delta kodları belirlemiştir ve yeni kodlara karşılık gelen tanımları Şekil 6'daki DT modeli XML dosyasına eklemiştir. Buradaki değişiklik DT konfigürasyonuna eklenen *cpu1* düğümü içindir. Uygulama burada, *cpu1* düğümünü fark olarak bulmakta ve XML kodunda da *cpus* düğümü altına *cpu0* düğümünün kardeşi olarak uygun yere *cpu1* düğümünü yerleştirmektedir. Bu işlem sonucunda DT modeli XML dosyası Şekil 7'deki gibi olmaktadır. Dosyada koyu gösterilen kısımlar uygulama tarafından eklenmiştir.

```
01  <devicetree_viewpoint:devicetree>
02    <root compatible="&quot;fsl,imx6dl-sabresd&quot;, &quot;fsl,imx6dl&quot;"
03        model="Kentkart i.MX6 DualLite Smart Device Board" name="root">
04      <cpus address_cells="1" name="cpus" size_cells="0">
05          <cpu0 clock_latency="61036" clock_names="&quot;arm&quot;,
06              &quot;pll2_pfd2_396m&quot;, &quot;step&quot;,&quot;pll1_sw&quot;,
07              &quot;pll1_sys&quot;, &quot;pll1_bypass&quot;, &quot;pll1&quot;,
08              &quot;pll1_bypass_src&quot;" compatible="&quot;arm,cortex-a9&quot;"
09              device_type="&quot;cpu&quot;" name="cpu0"
10              next_level_cache="&amp;L2"
11              operating_points="996000 1275000 792000 1175000 396000 1150000"
12              reg="0" soc_operating_points="996000 1175000 792000 1175000 396000
13                  1175000"/>
14          <cpu1 clock_latency="" clock_names="" clocks="" compatible="&quot;arm,
15              cortex-a9&quot;" device_type="&quot;cpu&quot;" name="cpu1"
16              next_level_cache="&amp;L2" operating_points="" reg="1"
17              soc_operating_points=""/>
18      </cpus>
```

Şekil 7. Uygulama tarafından değiştirilmiş DT modeli XML dosyasından bir kısım



Şekil 8. Araç sürücü terminali DT yapısı Core modelinin değişiklik sonrası görünümü

Şekil 7'de bir kısmı verilen DT modeli artık manuel yapılan değişiklikleri de içerecek şekilde güncellenmiştir. İlgili XML dosyası DSML4DT IDE'si içerisinde açıldığında güncelleme görsel modelde de görülebilmektedir. Şekil 8'de verilen bu yeni modelde artık *cpu1* DT düğümü de yer almaktadır. Böylece DT kodunda sonradan yapılan değişikliklerin DT modeline başarılı bir şekilde yantısıldığı ve sistem kodları ile modelin senkronize edilebildiği görülmüştür.

5 İlgili Çalışmalar

Literatürde DT'ler ve DT'lere bağlı sistem konfigürasyonlarının kullanıldığı çalışmalar bulunmaktadır. Örneğin donanım konfigürasyonlarının cihaz kayıtçılarının program içeriğinden ayrıştırılması [3], BTFRS gibi Linux dosya sistemleri dahilinde çoklu cihaz desteğinin sağlanması [4], heterojen Nesnelerin İnterneti ağlarında geçit görevi üstlenecek bazı anakart destek paketlerinin mikroişleyiciler için optimizasyonu [5], bulut hesaplama sistemlerindeki sanallaştırılma katmanları dahilinde bilgisayar fiziksel belleklerinin misafir işletim sistemleri tarafından belirlenmesi [7], trafik işareti tanımlamalarında kullanılacak gömülü sistemlerin inşaası [8] ve gerçek-zamanlı sağlık takibi cihazlarının üretilmesi [9] amacıyla DT yazılımları geliştirilmiştir. Ancak bu çalışmalar birer DT uygulaması örneğidir ve DT yazılımı modellemeyi ve/veya otomatik DT kodu üretimini göz önünde bulundurmamaktadır.

"Altera SoC EDS" isimli üründe [13] bir DT derleyicisi yer almakla birlikte kod üretimi sadece tek bir Linux çekirdeği için mümkündür. Neuendorffer [14] model tabanlı yaklaşımların Alan Programlanabilir Kapı Dizileri (FPGA) içindeki sistemlerin tasarlanmasında kullanılmasından bahsetmiş ve sistem tasarımlarından DT'lerin oluşturulabileceğini belirtmiştir. İlgili tasarım akışı, sunulan bir durum çalışması üzerinden örneklenmekle birlikte DT yapılarının otomatik oluşturma süreci çalışmada belli değildir. Jassi vd. [15] yazılım uygulamalarının Linux üzerinde çalıştırılması durumunda geliştirdikleri GRIP isimli aracın DT kaynak dosyalarını da üretmesinin mümkün olduğunu belirtmişlerdir ancak bununla ilgili bir uygulama örneğine çalışmalarında rastlanmamaktadır. Bu bildiride tanıtılan tersine mühendislik yaklaşımı ile literatürdeki bu kısıtlı sayıdaki DT yazılımlarının otomatik üretilme ve model-güdümlü geliştirilmesi çalışmalarına üretilen kodlardaki değişikliklerin DT yazılım modellerine yansıtılması yönüyle bir katkı verildiğine inanılmaktadır.

Bildiğimiz kadarıyla mevcut çalışmalardan hiçbiri DT kodları ve modelleri arasındaki senkronizasyonu sağlamak için bir yöntem sunmamaktadır.

6 Değerlendirme ve Sonuç

Bu çalışmada, DT yazılımlarının model-güdümlü geliştirilmesi sırasında mevcut yazılım kodlarından DT modellerinin otomatik olarak geri elde edilmesini sağlayacak bir tersine mühendislik yöntemi tanıtılmıştır. Bu yöntemin DSML4DT adlı bir DSML ve bu dile ait IDE ile kullanılması amacıyla hazırlanan bir uygulamanın örneklenmesi de endüstriyel bir durum çalışması üzerinden gösterilmiştir. Bildiride tanıtılan uygulama kullanılarak DT yazılımı kaynak kodlarının ayrıştırılması yapılmakta ve DSML4DT ile uyumlu görsel DT modellerine ait tasarımları içeren XML dosyaları DT yazılımı kodlarında yapılan değişiklikleri içerecek şekilde otomatik olarak üretilmektedir. Böylece DT yazılımları ile bunların DSML4DT modelleri arasındaki senkronizasyon sağlanabilmektedir.

Bu çalışmada, detayları verilen araç sürücüsü terminali sistemi durum çalışmasına ek olarak Kentkart firmasında geliştirilmiş olan farklı gömülü sistem cihazları üzerinde çalışan çeşitli DT yazılımları için de DT kodlarındaki değişikliğin DSML4DT ile oluşturulan sistem modellerine yansıtılması sağlanmıştır. Değerlendirme amacıyla bu firma tarafından geliştirilen validatör (ücret düşüm) cihazı, araç içi kamera sistemi kontrol cihazı ve araç içi reklam görüntüleme cihazlarının hazır DT dosyalarından tersine mühendislik uygulaması ile DT görsel modellerinin sorunsuzca elde edilebildiği görülmüştür. Bununla birlikte bu çalışmada geliştirdiğimiz uygulama yine bu cihazlarda elle değiştirilmiş olan DT kodlarındaki değişiklikleri de bu cihazlara ait DSML4DT örnek DT modellerine başarılı bir şekilde yansıtabilmiştir. Bölüm 4'teki durum çalışmasında gösterilen işlemci düğümünden farklı olarak bellek, entegre arası devre (i2c), seri çevresel arayüz (spi), yüksek çözünürlüklü çoklu ortam arayüzü (hdmi) gibi birçok farklı DT düğümü için de geliştirdiğimiz uygulama kodlar ve görsel modeller arasındaki senkronizasyonu sağlayabilmektedir.

Kentkart firmasında yapılan örnek çalışmalarda yazılım geliştiricilerin DSML4DT kullanarak oluşturduğu DT yazılımlarında sonradan yaptıkları değişiklerin daha çok DT yapısına yeni düğüm eklemek ve/veya düğüm içlerindeki özniteliklerde güncellemelerde bulunmak olduğu tespit edilmiştir. Bu çalışmada tanıtılan uygulama bu değişiklikler için tersine mühendisliği tam olarak sağlamaktadır. Bununla birlikte bu bildirinin 3. Bölümünde tarif edilen 2. senaryodaki gibi daha önceden herhangi bir yazılım modeli bulunmayan DT dosyalarından da yazılım modellerinin elde edilebildiği görülmüştür. Ancak bu kodlardan daha önce DSML4DT kullanılmadan başka yöntemlerle üretilmiş olanlarının bir kısmında görsel modellere ait XML dosyalarının oluşturulmasında eksikliklerin olduğu gözlenmiştir. Bunun başlıca nedeni bu tip kodların DSML4DT bakış açılarına tam uyumlu olmayan tanımlamaları içermesi ve bunların ayrıştırıcı uygulamanın çalışması sırasında DSML4DT görsel somut sözdizimindeki karşılıklarının bulunamamasıdır. İleriye yönelik planlanan ilk çalışma bu yöndeki eksikliği tamamlayacak bazı dönüşüm mekanizmalarının

uygulamaya dahil edilmesidir. Ayrıca tersine mühendislik yaklaşımının başka endüstriyel DT yazılımı geliştirme çalışmalarında kullanılması ve daha geniş bir perspektife kullanımının değerlendirilmesi çalışmaları yürütülecektir.

Teşekkür

Bu çalışma TÜBİTAK ARDEB-EEEAG tarafından desteklenen 117E553 no'lu "Aygıt Ağacı Yazılımlarının Farklı Gömülü Sistem Platformları için Model Güdümlü Geliştirilmesi" başlıklı proje kapsamında gerçekleştirilmiştir.

Kaynakça

1. The Devicetree Specification, https://www.devicetree.org/, last accessed 2019/08/19.
2. Madieu, J.: The Concept of a Device Tree, 139-166. Linux Device Drivers Development: Develop customized drivers for embedded Linux. Packt Publishing (2017).
3. Schüpbach, A., Baumann, A., Roscoe, T., Peter, S.: A Declarative Language Approach to Device Configuration. ACM Transactions on Computer Systems, 30(1), Article 5 (2012).
4. Rodeh, O., Bacik, J., Mason, C.: BTRFS: The Linux B-Tree Filesystem. ACM Transactions on Storage, 9(3), Article 9: 1-32 (2013).
5. Gioia, E., Passaro, P., Petracca, M.: AMBER: An advanced gateway solution to support heterogeneous IoT technologies. In proceedings of the 24th International Conference on Software, Telecommunications and Computer Networks, 1-5 (2016).
6. Arslan, S., Türk, E. Kardaş, G.: Gömülü Sistem Yazılımlarının Geliştirilmesinde Aygıt Ağacı Yapısının Kullanılmasına Yönelik bir Çalışma. 2. Uluslararası Bilgisayar Bilimleri ve Mühendisliği Konferansı, IEEE Conference Publications, 882-887 (2017).
7. Bielski, M., Rigo, A., Pacalet, R.: Dynamic Guest Memory Resizing - Paravirtualized Approach. In proceedings of the 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, 181-186 (2019).
8. Farhat, W, Sghaier, S, Faiedh, H, Souani, C: Design of efficient embedded system for road sign recognition. Journal of Ambient Intelligence and Humanized Computing, 10(2), 491-507 (2019)
9. Swaroop, K. N., Chandu, K., Gorrepotu, R., Deb, S.: A health monitoring system for vital signs using IoT. Internet of Things, 5, 116-129 (2019).
10. Arslan, S., Kardaş, G.: Aygıt Ağacı Yazılımlarının Modellenmesi, 12. Ulusal Yazılım Mühendisliği Sempozyumu, CEUR Workshop Proceedings, 2201: 1-12 (2018).
11. Canfora, G., Di Penta, M., Cerulo, L.: Achievements and challenges in software reverse engineering. Communications of the ACM 54(4): 142-151 (2011).
12. Washizaki, H., Gueheneuc, Y.-G., Khomh, F.: ProMeTA: a taxonomy for program metamodels in program reverse engineering. Empirical Software Engineering 23(4): 2323-2358 (2018).
13. Rocketboards: Golden System Reference Design, https://rocketboards.org/foswiki/view/Documentation/DeviceTreeGenerator, last accessed 2019/08/19.
14. Neuendorffer, S.: FPGA Platforms for Embedded Systems. 351-379. Model-Based Design for Embedded Systems. Nicolescu, G., Mosterman P. J. (Eds.), CRC Press (2018).
15. Jassi, M., Hu, Y., Mueller-Gritschneder, D., Schlichtmann, U.: Graph-Grammar-Based IP Integration (GRIP)—An EDA Tool for Software-Defined SoCs. ACM Transactions on Design Automation of Electronic Systems, 23(3), Article 40: 1-26 (2018).

Meta-Modelleme Araçlarının Uygulayıcıların İhtiyaçları Açısından Analizi

Mert Ozkaya[1] ve Deniz Akdur[2]

[1] Yeditepe Üniversitesi, Bilgisayar Mühendisliği Bölümü, İstanbul, Türkiye
mozkaya@cse.yeditepe.edu.tr
[2] Aselsan A.Ş., Ankara
denizakdur@aselsan.com.tr

Özet. Günümüzde uygulayıcıların alana-özgü modelleme dili ve ilgili araçları (modelleme editörü ve kod dönüştürücüler gibi) geliştirmelerine imkan sağlayan birçok farklı meta-modelleme aracı bulunmaktadır. Fakat, uygulayıcıların mevcut meta-modelleme araçları arasında kendi gereksinimlerine en uygun olanları tes-pit edip karşılaştırma yapabilmesi hali hazırda pek de kolay gözükmemektedir. Bundan dolayı, bu çalışmada, mevcut 20 farklı meta-modelleme aracı farklı endüstri-lerde çalışan uygulayıcıların ilgisini çekebileceğini düşündüğümüz bir küme gerek-sinim açısından analiz edilmiştir. Bu gereksinimler şöyledir: *(i)* satıcı kilidi, *(ii)* web üzerinden erişim, *(iii)* iş-birliği halinde modelleme, *(iv)* modellerden versiyonların oluşturulması, *(v)* hibrit modelleme, *(vi)* model dönüşümü, *(vii)* harici araç desteği, *(viii)* topluluk desteği, *(ix)* genişletilebilirlik, ve *(x)* dil test edilebilirliği. Bu çalışmanın sonuçlarının, uygulayıcıların alana-özgü modelleme dili geliştirmek için kullanmaları gereken meta-modelleme araçlarını karşı-laştırıp kendileri için en uygun aracı seçmeleri konusunda fazlaca yardımcı olacağı düşü-nülmektedir. Sonuçların ayrıca meta-modelleme aracı geliştiricileri için de eksikliklerin tespit edilip düzeltilebilmesi konusunda yardımcı olması beklenmektedir.

Anahtar Kelimeler: Meta-modelleme, Alana-özgü diller, Meta-modelleme araçları, Literatür analizi

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Towards Analysing the Meta-Modeling Tools for the Needs of Practitioners

Mert Ozkaya[1] and Deniz Akdur[2]

[1] Yeditepe University, Computer Engineering Department, Istanbul, Turkey
mozkaya@cse.yeditepe.edu.tr
[2] Aselsan A.Ş., Ankara
denizakdur@aselsan.com.tr

Abstract. Many meta-modeling tools have been available today for the use of practitioners and enabling them to develop domain-specific languages (DSLs) and the necessary toolset (i.e., modeling editors and code generators) for performing the modeling activities in their domain. However, it is not currently so easy for practitioners to compare the existing meta-modeling tools for their practical needs and choose the most suitable one(s). So, in this paper, 20 different meta-modeling tools have been intended to be analysed for a number of requirements that are believed to be important for the practitioners working in different industries. These requirements are *(i)* vendor lock-in, *(ii)* web-access, *(iii)* collaborative modeling, *(iv)* model versioning, *(v)* hybrid modeling, *(vi)* model transformation *(vii)* external tool integration, *(viii)* community support, *(ix)* extensibility, and *(x)* language testability. The analysis results reveal many important findings about the meta-modeling tools and will be expected to be very helpful for the practitioners in determining any meta-modeling tool(s) that best suit their needs for their language development. Also, the tool vendors may use the results to find out the weakness of the existing tools and improve them in the future.

Keywords: Meta-modeling, Domain-specific languages, Meta-modeling tools, Literature analysis

1 Introduction

With the continuous evolution of technology, software systems have nowadays been penetrating all areas of the society to reduce the efforts required by people in any sorts of tasks that they perform [1]. This essentially requires the software systems to reach an ever-increasing level of complexity that makes them harder and harder to manage. Indeed, cyber-physical systems in which the software systems are expected to control the physical devices (e.g., cars, trains, planes, smart cities, etc.) are good examples of these complex software systems whose behaviours may change at any time depending on their heterogeneous environment [10]. One effective way of managing the software complexity herein is the abstraction, which promotes focusing on the actual problem(s) and suppressing the irrelevant details [6]. The model-driven software development (MDSD)

paradigm [12] has been proposed in the early two-thousands with the purpose of creating abstract models that each address a particular concern of the software systems to be built without getting into any programming language details.

As indicated by Schmidt [11], MDSD can essentially be considered as the combinations of the *(i)* domain-specific languages (DSLs) and *(ii)* transformation engines and generators. DSLs offer domain-specific notation sets for the modeling of software systems in particular domains (e.g., automative, avionics, embedded systems, and multi-agent systems). The transformation engines and generators provide the the tool support for processing the software models specified with DSLs for various purposes such as formal verification, software code generation, model-based testing, and model refactoring.

To create DSLs, the meta-modeling tools can be used. The meta-modeling tools allow for defining the meta-model of the DSLs, which comprises the definitions of the language syntax and semantics that describe the language constructs, their relationships, abstract syntax tree, and any constraints and rules on them. The meta-modeling tools may also offer various facilities such as code generator definition language, versioning, multi-user support, and language testability.

Many different meta-modeling tools are available for use by the language developers. However, the current literature on the meta-modeling tools suffers from a number of issues that prevent practitioners from determining the meta-modeling tools that best satisfy their expectations for the language development and comparing the existing tools for the practical concerns such as versioning, testability, multi-user collaboration, hybrid modeling, external tool integration, etc. Therefore, in this paper, the existing meta-modeling tools are aimed to be analysed for a set of requirements that are believed to be highly important for the practitioners in industry who are willing to use any meta-modeling tools.

2 Research Methodology

To determine the existing meta-modeling tools to be analysed, the google search engine has been used and several keywords have been searched on google, including "DSL tools", "meta-modeling tools", "meta-modeling languages", "DSL development", "language engineering", and "language engineering tools". Also, the web-site of each meta-modeling tool has been checked to determine whether the web-sites include any tool reviews. Indeed, Metaedit+'s web-site has a comprehensive tool review section. As a result, 20 different meta-modeling tools have been identified, whose details are given in Table 1. So, while some meta-modeling tools are provided with the installations on the Windows, MacOS, and Linux platforms, some tools are provided with the Eclipse update-site and run on the Eclipse platform only. Also, most of the meta-modeling tools are open-source and allow the users to modify the tools for their particular needs (if necessary).

To determine the set of requirements for which the meta-modeling tools are to be analysed, we used the results of our recent empirical researches that focus on the practitioners' experiences on software modeling languages and the analysis of the modeling tools (e.g., [7,8]). Moreover, thanks to our relationships with

industry and involvement in many research projects concerning both academia and industry, the authors received many valuable feedback from several practitioners who work in diverse industries (including finance, defense & military, and automotive & transportation) regarding their expectations from the meta-modeling tools. So, the following set of requirements has been determined:

Table 1. The meta-modeling tools for creating DSLs

Meta-Modeling Tools	URL	Platform	Open Source
Metaedit+	www.metacase.com	Windows, Linux, Mac	No
MPS	www.jetbrains.com/mps/	Windows, Linux, Mac	Yes
Sirius	www.eclipse.org/sirius/	Eclipse Plugin	Yes
Xtext	www.eclipse.org/Xtext/	Eclipse Plugin	Yes
Graphiti	www.eclipse.org/graphiti/	Eclipse Plugin	Yes
Spoofax	www.metaborg.org	Eclipse and IntelliJ Plugins, Jar File	Yes
Diagen	www2.cs.unibw.de/tools/DiaGen/	Windows, Linux, Mac	Yes
AToMPM	atompm.github.io	Windows, Linux, Mac	No
Fujaba (E-Moflon)	web.cs.upb.de/archive/fujaba/	Windows, Linux, Mac	Yes
VMTS	vmts.aut.bme.hu	Windows, Linux, Mac	No
GEMS	wiki.eclipse.org/GEMS	Eclipse Plugin	Yes
GME	www.isis.vanderbilt.edu/Projects/gme/	Windows	Yes
MS Visual Studio	visualstudio.microsoft.com	Windows, Linux, Mac	No
MetaModelAgent	www.metamodelagent.com/	Eclipse Plugin and RSA Plugin	Yes
ConceptBase	conceptbase.sourceforge.net	Windows, Linux, Mac	Yes
MetaDepth	metadepth.org	Windows, Linux, Mac	No
ADOXX	www.adoxx.org	Windows and Mac	No
JastEMF	bitbucket.org/jastemf/jastemf-plugins/wiki/	Eclipse Plugin	No
Melange	http://melange.inria.fr	Eclipse Plugin	Yes
WebGME	webgme.org	Windows, Linux, Mac	No

Vendor Lock-in. Vendor lock-in is concerned with the meta-modeling tool's capability of getting the DSLs to be produced that can be used independently from the meta-modeling tool. By doing so, the DSLs may also be imported into some other meta-modeling tools that also suit the needs of practitioners and practitioners will not be restricted with a single meta-modeling tool.

Web-access. Web-access is concerned with the meta-modeling tool's capability of being used over internet via some browsers without the need for downloading and installing any applications. So, practitioners may perform their meta-modeling activities or use any DSLs for modeling over internet.

Collaborative Modeling. Collaborative modeling is considered in terms of concurrent meta-modeling and working offline. Concurrent meta-modeling is to do with allowing multiple users to work on the same meta-model (or DSL for modeling) simultaneously. Working offline is to do with multiple users modifying the same meta-model (or model via DSL) offline and getting their changes to be reflected once connected to the internet automatically.

Model Versioning. Model versioning is concerned with the meta-modeling tool's capability of storing and managing different versions of the meta-models or the models created with the DSLs of that meta-modeling tool. The versioning support is considered in terms of the following requirements: *(i)* repository support, *(ii)* tracking versions, *(iii)* comparing versions, *(iv)* filtering versions, *(v)* merging versions, and *(vi)* secure versioning.

Hybrid Modeling. Hybrid modeling is concerned with the meta-modeling tool's capability of enabling the textual and visual representations of the lan-

guage constructs that can be used together in a synchronised manner for the hybrid modeling of software systems.

Model Transformation. Model transformation is concerned with the meta-modeling tool's capability of transforming a model into another model for different purposes such as software implementation and formal verification.

External Tool Integration. External tool integration is concerned with the meta-modeling tool's capability of enabling the integration with other external tools for performing some useful facilities that are not possible with the meta-modeling tool (e.g., combining model and code together, simulation, analysis, and code-generation).

Community Support. The community support is concerned with the tools' capability of providing users with the necessary materials and platforms that can be useful for the users in learning and using the meta-modeling tools. The community support is considered in terms of the *(i)* tutorials, *(ii)* forum, *(iii)* e-mail support, *(iv)* user-manual, *(v)* training sessions, and *(vi)* consulting team.

Extensibility. Extensibility is concerned with the meta-modeling tool's capability of getting extended with some additional requirements that practitioners need in their domain such as the model analysis, versioning, multi-user support, code-generation, etc.

Language Testability. Testability is concerned with the meta-modeling tool's capability of testing the DSLs created via the meta-modeling tool and checking their correctness. Testability is considered in terms of the *(i)* language definition (syntax and semantics) testing, *(ii)* language editor testing, and *(iii)* model transformation testing.

3 Analysis Results

In this section, the analysis results of the 20 different meta-modeling tools for each requirement considered are discussed separately. The entire analysis results can be found as a whole in the project web-site[3].

3.1 Vendor Lock-in

As Table 2 shows, many meta-modeling tools (60%) require the vendor lock-in and force practitioners to use the same vendor's meta-modeling tool for developing the DSLs and using those DSLs for specifying models. However, a few tools (Sirius, Xtext, Graphiti, GEMS, JastEMF, and Melange) are based on the Eclipse's EMF framework for the definition of meta-models, and thus the produced EMF models in those meta-modeling tools can be accepted by any tools that support EMF. Also, Metaedit+ and ADOXX export models and meta-models of the languages in the XML format; so, any tools that accept XML can process the models and meta-models created with Metaedit+ or ADOXX.

[3] The entire analysis results for the meta-modeling tools can be found in `https://sites.google.com/site/drmertozkaya/metamodelingtools`

Table 2. The meta-modeling tools with vendor lock-in support

DSL Tools	Vendor Lock-in
Metaedit+	No - Import/export XML
MPS	Yes
Sirius	No - EMF
Xtext	No - EMF
Graphiti	No - EMF
Spoofax	Yes
Diagen	Yes
AToMPM	Yes
Fujaba (E-Moflon)	Yes
VMTS	Yes
GEMS	No - EMF
GME	Yes
MS Visual Studio	Yes
MetaModelAgent	Yes
ConceptBase	Yes
MetaDepth	Yes
ADOXX	No - import/export XML
JastEMF	No - EMF
Melange	No - EMF
WebGME	Yes

Table 3. The meta-modeling tools that are accessible over internet

DSL Tools	Web Support
Metaedit+	No
MPS	No
Sirius	No
Xtext	Yes
Graphiti	No
Spoofax	No
Diagen	No
AToMPM	Yes
Fujaba (E-Moflon)	No
VMTS	No
GEMS	No
GME	No
MS Visual Studio	No
MetaModelAgent	Yes
ConceptBase	No
MetaDepth	No
ADOXX	No
JastEMF	No
Melange	No
WebGME	Yes

3.2 Web-access

As Table 3 shows, only four meta-modeling tools provide web-access support. These are AtomPM, MetaModelAgent, WebGME, and Xtext. AtomPM herein provides the most comprehensive support, enabling the users to access the meta-modeling tool via browser, terminal-based client, and smartphones for the meta-modeling and modeling. WebGME and Xtext enable the browser-access for the meta-modeling and modeling activities. MetaModelAgent however supports browsers basically for communicating the models created over internet without allowing for accessing and editing the meta-models.

Table 4. The meta-modeling tools that support model versioning

DSL Tools	Repository	Tracking Versions	Comparing Versions	Filtering Versions	Merging Versions	Secure Versioning
Metaedit+	GIT, SVN, Server-based Repository	Yes	Yes	Yes	No	No
MPS	GIT	Yes	Yes	No	Yes	No
Sirius	GIT, CVS, SVN, CDO-based Repository	Yes	Yes	No	Yes	Authentication and data-access rights
Xtext	No	No	No	No		
Graphiti	No	No	No	No		
Spoofax	No	No	No	No		
Diagen	SVN	No	No	No		
AToMPM	No	No	No	No		
Fujaba (E-Moflon)	No	No	No	No		
VMTS	No	No	No	No		
GEMS	No	No	No	No		
GME	Server-based	No	No	No	No	No
MS Visual Studio	No	No	No	No		
MetaModelAgent	No	No	No	No		
ConceptBase	No	No	No	No		
MetaDepth	No	No	No	No		
ADOXX	No	No	No	No		
JastEMF	No	No	No	No		
Melange	No	No	No	No		
WebGME	Cloud-based	Yes	Yes	No	Yes	Authentication

3.3 Model Versioning

As Table 4 shows, Metaedit+, MPS, Sirius, GME, and WebGME support the versioning of the meta-models of the DSLs and the models specified using the DSLs. Metaedit+ supports the integration with the external versioning tools, such as GIT, SVN, and at the same time offers their own server-based repositories. While Metaedit+ enables tracking, comparing, and filtering versions, other important requirements such as merging different versions in the same (meta-)model and the secure access to the model versions are neglected in Metaedit+. MPS supports GIT and just enables tracking, comparing, and merging model versions. While Sirius itself does not support versioning, the ObeoDesigner tool developed by the Obeo company for Sirius supports various external versioning tools (i.e., GIT, CVS, and SVN) and a server-based repository that enables other useful operations including tracking, comparing, and merging versions. Also, ObeoDesigner supports the authentication and secure data-access for the model versions. GME supports a server-based repository for storing and accessing the model versions. WebGME (i.e., the extension of GME) provides more comprehensive support, offering a cloud-based repository system that allows for tracking, comparing, and merging versions and their authenticated access.

3.4 Collaborative Modeling

As in Table 5, Metaedit+, MPS, Sirius, GEMS, and WebGME are the only meta-modeling tools that support collaboration. Metaedit+, MPS, GEMS, and WebGME allow for the concurrent access only. Sirius essentially supports both concurrent access and working offline via the Obeo Designer platform.

3.5 Hybrid Modeling

As in Table 6, MPS, Sirius, JastEMF, and Melange are the only meta-modeling tools that support the hybrid modeling. MPS focuses on storing the abstract syntax tree (AST) of a model created, and those ASTs can be accessed via the projectional editors that use any projections defined over ASTs (i.e., the textual or visual representations of the ASTs). Sirius supports the integration with the Xtext meta-modeling tool. While Sirius is a graphical meta-modeling tool, Xtext is a textual one. Also, both of them are Eclipse frameworks that are based on EMF. So, when they are integrated, the resulting meta-modeling tool can be used for defining a language for hybrid modeling. JastEMF is also based on EMF and integrates the JastAdd meta-compilation system for specifying the executable semantics of the EMF models. The EMF models herein can be the textual Xtext or the Eclipse Graphical Modeling Framework (GMF). However, JastEMF does not provide support for the hybrid modeling where the textual and visual representations can be used together on the same model. Melange focuses essentially on extending and re-using the existing DSLs that are based on EMF. So with Melange, it is actually possible to combine a textual DSL with another graphical DSL for enabling the hybrid modeling.

Table 5. The meta-modeling tools that support the collaborative modeling

DSL Tools	Concurrent Access	Work Offline
Metaedit+	Yes	No
MPS	Yes	No
Sirius	Yes	Yes
Xtext	No	No
Graphiti	No	No
Spoofax	No	No
Diagen	No	No
AToMPM	Yes	No
Fujaba (E-Moflon)	No	No
VMTS	No	No
GEMS	Yes	No
GME	No	No
MS Visual Studio	No	No
MetaModelAgent	No	No
ConceptBase	No	No
MetaDepth	No	No
ADOXX	No	No
JastEMF	No	No
Melange	No	No
WebGME	Yes	No

Table 6. The meta-modeling tools that support hybrid modeling

DSL Tools	Visual Notation Set	Textual Notation Set
Metaedit+	Yes	No
MPS	Yes	Yes
Sirius	Yes	Yes
Xtext	No	Yes
Graphiti	Yes	No
Spoofax	No	Yes
Diagen	Yes	No
AToMPM	Yes	Yes
Fujaba (E-Moflon)	Yes	No
VMTS	Yes	No
GEMS	Yes	No
GME	Yes	No
MS Visual Studio	Yes	No
MetaModelAgent	Yes	No
ConceptBase	Yes	No
MetaDepth	No	Yes
ADOXX	Yes	No
JastEMF	Yes	Yes
Melange	Yes	Yes
WebGME	Yes	No

3.6 Model Transformation

As in Table 7, most meta-modeling tools are supported with a transformation language that enables for developing a translator which walks through the abstract syntax trees of the models created. Graphiti, Diagen, GEMS, MetaModelAgent, and ConceptBase are the tools that neglect the model transformation.

Table 7. The meta-modeling tools that support model transformation

DSL Tools	Model Transformation
Metaedit+	MERL Metaedit Reporting Language
MPS	A Generator Definition Language
Sirius	Accelelo Query Language
Xtext	Xtend Language
Graphiti	No
Spoofax	Stategо Transformation Language
Diagen	No
AToMPM	HOT rules
Fujaba (E-Moflon)	Triple Graph Grammar
VMTS	API
GEMS	No
GME	Graph Rewriting
MS Visual Studio	T4 Framework
MetaModelAgent	No
ConceptBase	No
MetaDepth	Epsilon Languages
ADOXX	AdoScript
JastEMF	JastADD
Melange	Xtend Language
WebGME	JavaScript Language

Table 8. The meta-modeling tools that support the tool integration

DSL Tools	External Tool Integration
Metaedit+	ASCII/XML/SOAP, Visual Studio, Simulink
MPS	IntelliJ
Sirius	Eclipse Xtext
Xtext	GEF, Sirius, Graphiti
Graphiti	No
Spoofax	No
Diagen	No
AToMPM	No
Fujaba (E-Moflon)	No
VMTS	No
GEMS	No
GME	COM Components
MS Visual Studio	No
MetaModelAgent	No
ConceptBase	No
MetaDepth	No
ADOXX	Vendor-specific tools and services
JastEMF	Vendor-specific tools and services
Melange	No
WebGME	Vendor-specific tools and services

3.7 External Tool Integration

As Table 8 shows, 40% of the meta-modeling tools support the external tool integration. Among them, Metaedit+ and MPS support the round-trip engineering via the external tools. Metaedit+ can be integrated with Visual Studio and MPS

can be integrated with IntelliJ for synchronising the model with code. As aforementioned, Sirius can be integrated with Xtext to enable the development of a hybrid modeling language. GME allows for writing plug-ins in any programming languages that support the COM standard. Lastly, ADOXX, JastEMF, and WebGME offer a set of built-in tools and services that can be integrated.

3.8 Community Support

As shown in Table 9, while most of the meta-modeling tools provide tutorials or user-manuals, the rest of the requirements considered are not so popular. 60% of the meta-modeling tools provide e-mail addresses or mailing lists through which the users can get any help. The discussion platforms such as forum are even less supported. The least support has been provided for the training sessions and consulting teams, supported by 4 DSL tools only.

3.9 Extensibility

As Table 10 shows, 40% of the DSL tools support extensibility. While some of those DSL tools allow users to develop their own extensions via an API support, some just offer pre-built extensions that can be plugged-in by the users. Note that ADOXX and Melange are the exceptions herein. ADOXX supports making a remote-call to the external services to be able to exploit their functionalities. Melange is essentially a language workbench that enables to design a new DSL by re-using the syntax and semantics of the existing DSLs and their tools.

Table 9. The meta-modeling tools that provide community support

DSL Tools	Tutorial	Forum	E-mail	User man.	Training	Consultancy
Metaedit+	X	X	X	X	X	X
MPS	X	X	X	X	X	X
Sirius	X	X		X		
Xtext	X	X		X		
Graphiti	X		X	X		
Spoofax	X		X	X		
Diagen	X		X	X		
AToMPM	X			X		
Fujaba (E-Moflon)			X	X		
VMTS			X	X		
GEMS	X					
GME	X		X	X		
MS Visual Studio		X		X		
MetaModel-Agent	X	X	X	X	X	X
ConceptBase	X	X	X	X		
MetaDepth	X		X	X		
ADOXX	X	X	X	X	X	X
JastEMF						
Melange	X			X		
WebGME	X			X		

Table 10. The meta-modeling tools that support extensibility

DSL Tools	Extensibility
Metaedit+	No
MPS	User-defined
Sirius	No
Xtext	User-defined
Graphiti	No
Spoofax	No
Diagen	No
AToMPM	No
Fujaba (E-Moflon)	No
VMTS	Pre-build, user-defined
GEMS	No
GME	User-defined plug-in
MS Visual Studio	Pre-built, user-defined
MetaModelAgent	No
ConceptBase	
MetaDepth	No
ADOXX	Call to external applications
JastEMF	No
Melange	Re-using existing DSLs
WebGME	Pre-built

3.10 Testability

The analysis results show that only MPS, Spoofax, and VMTS support testability. While MPS and VMTS allow for testing the language definition, transformations, and editors, Spoofax allows for testing the language definition and transformation only.

4 Related Work

While the literature includes some analytical studies for the meta-modeling tools, none of them consider as many meta-modeling tools as considered in this paper. Moreover, the existing analytical studies do not focus on some of the practical requirements that are considered in this paper, such as the hybrid modeling, collaboration, community support, web-access, and vendor lock-in.

In [9], the author analysed three meta-modeling tools for a set of requirements including documentation support, abstract syntax, concrete syntax, and semantics. In [13], the author analysed the Microsoft meta-modeling tools (e.g., Visual Studio) and Eclipse-based meta-modeling tools for a set of requriements including the meta-modeling features, repository, graphical notation, model-to-model transformation, model-to-text transformation, and model validation. In [4], the author compared Eclipse GEF with Metaedit+ for their usabilities, e.g., the time and effort needed to create DSLs, graphical meta-modeling, and separation of concerns. In [3], the author analysed the code generation facilities of 10 meta-modeling tools in terms of a set of requirements, which are the functional requirements, reliability, maintainability, extendibility, improvability, portability, usability, and financial saving. In [2], the authors analysed 5 meta-modeling tools via a case-study that focuses on creating a language and toolset for drawing BPMN diagram. The authors considered a set of requirements in their comparisons including the graphical expressiveness & completeness, tool openness, and tool usability. In [5], the authors analysed 6 meta-modeling tools in terms of their support for a set of meta-modeling concepts. These concepts are object type, relationship type, role type, port type, model type, links to model types, attributes, inheritance, grouping, identification, and constraint language.

5 Conclusion

20 meta-modeling tools have been analysed in this paper for a set of requirements that are believed to be important for practitioners: *(i)* vendor lock-in, *(ii)* web-access, *(iii)* collaborative modeling, *(iv)* model versioning, *(v)* hybrid modeling, *(vi)* model transformation *(vii)* external tool integration, *(viii)* community support, *(ix)* extensibility, and *(x)* language testability. So, none of the meta-modeling tools fully support all those requirements at the same time.

60% of the meta-modeling tools require vendor lock-in and force practitioners to use the same meta-modeling tool for the (meta-)modeling activities. Concerning the rest, most are based on EMF and enable the (meta-)models created in

one EMF-based tool to be used in another EMF-based tool to some extent. While web-access is also a crucial requirement for avoiding the efforts on downloading and installing any applications, most of the meta-modeling tools are available as desktop applications only. Xtext, AToMPM, and WebGME are the only meta-modeling tools that can be accessed over web-browsers. Note that AToMPM may further be accessed via the mobile applications. Just 6 meta-modeling tools support the collaborative (meta-)modeling. While all these tools enable the simultaneous access, the Sirius meta-modeling tool further supports the work-offline facility where practitioners do not need the internet connection to collaborate. Concerning the meta-model versioning, just 6 meta-modeling tools enable to keep a repository of different versions of the meta-models. Managing different versions, such as tracking, comparing, filtering, merging the versions, and secure versioning are not supported by all those 6 tools though. WebGME seems to be the only meta-modeling tools that indeed supports the version management in terms of all those requirements. Concerning the hybrid modeling support, while most of the meta-modeling tools support the graphical representations of the language constructs, the textual representations are not that popular. Indeed, MPS, Sirius, AToMPM, and Melange are the only meta-modeling tools that support both the graphical and textual representations of the language constructs that can be used for modeling in a synchronised manner. Model transformation is supported by 75% of the meta-modeling tools considered, each of which provide a generator definition language for practitioners to develop any code generators that they need for mapping the models created in their DSLs. Concerning the external tool integration, Metaedit+ and MPS are the meta-modeling tools that can be integrated with the programming environments such as Visual Studio and IntelliJ so as to combine modeling and coding together. While a few other meta-modeling tools enable the user-defined and built-in tools to be integrated, most of the meta-modeling tools considered ignore the external tool integration. While most meta-modeling tools' web-sites include tutorials, and some discussion platforms (e.g., forum, mailing list), just a few of them organise training sessions and provide consultancies on the tool usage. These tools are Metaedit+, MPS, MetaModelAgent, and JastEMF. Concerning extensibility, 40% of the meta-modeling tools support extending the tools with some new requirements, and among them, VMTS and Visual Studio provide both ready-to-use and user-defined extension plug-ins. The rest herein do not enable to define new extensions and rather use a pre-defined set. While testing the DSLs developed is highly important for checking the language definition, model editor, and model transformation for some desired properties, MPS, Spoofax, and VMTS are the three meta-modeling tools that support the language testability at varying levels.

In the future, the goal is to extend this work with a comprehensive set of requirements for which the meta-modeling tools will be analysed. This can be possible with a survey conducted on the practitioners or by following a well-established guidance on the meta-modeling tool requirements.

References

1. Afonso, M., Vogel, R., Teixeira, J.: From code centric to model centric software engineering: Practical case study of mdd infusion in a systems integration company. In: Proceedings of the Fourth Workshop on Model-Based Development of Computer-Based Systems and Third International Workshop on Model-Based Methodologies for Pervasive and Embedded Software. pp. 125–134. MBD-MOMPES '06, IEEE Computer Society, Washington, DC, USA (2006). https://doi.org/10.1109/MBD-MOMPES.2006.13, https://doi.org/10.1109/MBD-MOMPES.2006.13
2. El Kouhen, A., Dumoulin, C., Gérard, S., Boulet, P.: Evaluation of Modeling Tools Adaptation. Tech. rep. (Jan 2012), https://hal.archives-ouvertes.fr/hal-00706701
3. Farooji, F.S.: Evaluation of code generation tools. Tech. rep., Stockholm, Sweden (2014)
4. Kelly, S.: Comparison of eclipse emf/gef and metaedit+ for dsm. In: Proceedings of the 19th annual ACM conference on object-oriented programming, systems, languages, and applications, workshop on best practices for model driven software development (2004)
5. Kern, H., Hummel, A., Kühne, S.: Towards a comparative analysis of meta-metamodels. In: Proceedings of the Compilation of the Co-located Workshops on DSM'11, TMC'11, AGERE! 2011, AOOPES'11, NEAT'11, & VMIL'11. pp. 7–12. SPLASH '11 Workshops, ACM, New York, NY, USA (2011). https://doi.org/10.1145/2095050.2095053, http://doi.acm.org/10.1145/2095050.2095053
6. Krueger, C.W.: Software reuse. ACM Comput. Surv. **24**(2), 131–183 (Jun 1992). https://doi.org/10.1145/130844.130856, http://doi.acm.org/10.1145/130844.130856
7. Ozkaya, M.: Do the informal & formal software modeling notations satisfy practitioners for software architecture modeling? Information & Software Technology **95**, 15–33 (2018). https://doi.org/10.1016/j.infsof.2017.10.008, https://doi.org/10.1016/j.infsof.2017.10.008
8. Ozkaya, M.: Are the uml modeling tools powerful enough for practitioners? a literature review. IET Software (May 2019), https://digital-library.theiet.org/content/journals/10.1049/iet-sen.2018.5409
9. Philip De Smedt: Comparing three graphical DSL editors: AToM3, MetaEdit+ and Poseidon for DSLs. Tech. rep., University of Antwerp (2011), available from http://msdl.cs.mcgill.ca/people/hv/teaching/MSBDesign/201011/projects/Philip.DeSmedt/report/report_PhilipDeSmedt.pdf.
10. Rajkumar, R., Lee, I., Sha, L., Stankovic, J.: Cyber-physical systems: The next computing revolution. In: Design Automation Conference. pp. 731–736 (June 2010). https://doi.org/10.1145/1837274.1837461
11. Schmidt, D.C.: Guest editor's introduction: Model-driven engineering. Computer **39**(2), 25–31 (Feb 2006). https://doi.org/10.1109/MC.2006.58
12. Selic, B.: The pragmatics of model-driven development. IEEE Softw. **20**(5), 19–25 (Sep 2003). https://doi.org/10.1109/MS.2003.1231146, https://doi.org/10.1109/MS.2003.1231146
13. Turhan, O.: Comparison of Microsoft DSL Tools and Eclipse Modeling Frameworks for Domain-Specific Modeling in the context of Model-Driven Development. masters thesis, Department of Systems and Software Engineering, Blekinge Institute of Technology

Bir Yazılım Sisteminin Kullanımının Süreç Madenciliği Teknikleri ile Modellenmesi: Endüstriyel Bir Vaka Çalışması

Ümit Ülkem Yıldırım, Beytullah Sarıca, Ethem Utku Aktaş, and Mehmet Çağrı Çalpur

Softtech A.Ş., Arge Merkezi, 34947 İstanbul, Türkiye
{umit.yildirim,beytullah.sarica,utku.aktas,cagri.calpur}@softtech.com.tr

Özet. Süreç madenciliği, günümüz bilgi sistemlerinde yaygın olarak tutulan işlem günlükleri (log) kullanılarak, süreçlere ait bilginin otomatik olarak ortaya konulmasıdır. Bu yolla mevcut süreç modelleri keşfedilebildiği gibi, süreçlerin izlenmesi ve iyileştirilmesi de sağlanmaktadır. Ek olarak, bir yazılım sistemine ilişkin süreçlerin modeli, sistemde düzenli olarak yapılan iyileştirmeler ve değişiklikler nedeniyle, sürekli olarak değişmektedir. Dolayısıyla var olan yapının otomatik olarak ortaya konulabilmesi, sistem analistleri, mimarları ve yöneticileri için çok değerlidir. Bu çalışma ile bankacılık alanında ticari kredi başvuru yazılımı ile ilgili bir yazılım sisteminin süreç modelini sistem log verilerini kullanarak otomatik olarak çıkarmayı hedefledik. Bu amaçla, endüstriyel vakaya ait 800.000 adet web servis çağrısından elde edilen log verisini kullandık. Öncelikle, bu servis loglarını kullanarak, süreçleri ve aralarındaki geçişleri ortaya koyduk. Ortaya çıkan çıktı ile, yazılım uygulamasına ait süreç modelini elde ettik. Sonuç olarak, servis loglarından otomatik olarak süreç modellerinin elde edilmesini, yazılım sistem analistlerine, mimarlara ve diğer yöneticilere, süreçlerin izlenmesi ve iyileştirilmesi konusunda etkili bir yardımcı teknik olarak değerlendiriyoruz.

Anahtar Kelimeler: Log Analizi, Süreç Madenciliği, Servis Sistemleri, Yazılım Modelleme.

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Modelling the Use of a Software System with Process Mining Techniques: An Industrial Case Study

Ümit Ülkem Yıldırım, Beytullah Sarıca, Ethem Utku Aktaş, and Mehmet Çağrı Çalpur

Softtech Inc., Research and Development Center, 34947 Istanbul, Turkey
{umit.yildirim,beytullah.sarica,utku.aktas,cagri.calpur}@softtech.com.tr

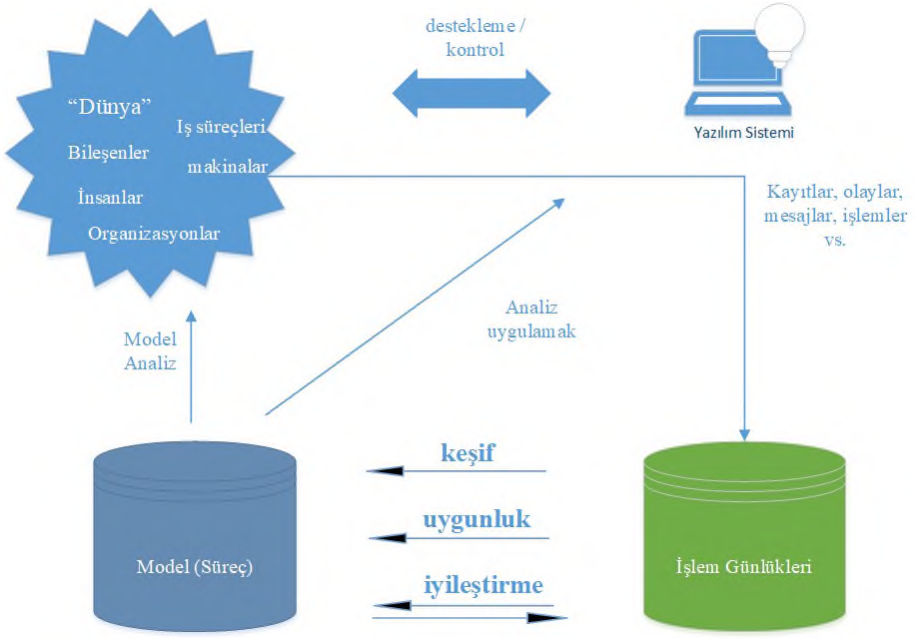
Abstract. Process mining is the automatic extraction of knowledge related to processes by using system logs which are commonly available in information systems. As well as discovering the current process models, they can also be used to monitor and improve the processes. In addition, the model of a software system may become very different from the initial, because of continuous improvements and changes in the system. Therefore, it is very valuable for system analysts, architects and managers to be able to reveal the existing structure automatically. In this study, we aim to automatically extract the process model of a software system related to commercial credit application software in banking domain, using system log data. For this purpose, we collected 800,000 web-service logs in our industrial case. Using these logs of web-services, we first extracted the processes and the transitions between them. Using this data, we obtained the process model of the application. As a result, we consider that automatically obtaining the process models from web-service logs is an effective auxiliary technique for the software system analysts, architects and other managers to monitor and improve their processes.

Keywords: Log Analysis, Process Mining, Service Systems, Software Modelling.

1 Giriş

Yazılım sistemlerinde yapılan işlemler için işlem günlükleri (log) tutulmakta, bu günlüklerle işlemlerin akışı takip edilmekte veya hata alınması durumunda hata analizi yapılarak sorunlara hızlı bir şekilde müdahale edilebilmektedir. Bu günlükler oldukça değerli veri içermektedir, bir servise ait girdiler ve çıktıların içeriği, işlemin ne zaman yapıldığı, ne kadar sürdüğü ve hata alınması durumunda hata ile ilgili detay bilgilere işlem günlüklerinden ulaşılabilmektedir. Büyük sistemlerde her gün milyonlarca işlem günlüğü kaydı oluşturulabilmekte ve oluşan bu veri, sistemin sağlığı, süreç akışları, sistemin modellenmesi gibi çeşitli konularda otomatik olarak çıkarım yapma olanağı sunmaktadır.

Süreç madenciliği, işlem günlüklerinden süreçlerle ilgili bilgilerin ortaya konulmasıdır. Günümüzde, süreç madenciliği teknikleri kullanılarak çeşitli uygulama alanlarındaki süreçleri keşfetmek, gözlemlemek ve geliştirmek mümkündür, bu sayede yeni araçların ortaya çıkması sağlanır. İşlem günlükleri için keşif (discovery), uygunluk (conformance) ve iyileştirme (enhancement) teknikleri kullanılarak süreç madenciliği yapılabilir (Şekil 1). Keşif tekniği herhangi başka bir bilgiye bakılmadan, işlem günlüklerini kullanarak bir model üretilmesidir. Uygunluk tekniğinde, mevcut bir süreç modeli aynı işlemin olay günlüğüyle karşılaştırılır. İyileştirme tekniğinde ise olay günlüklerine kaydedilen asıl işlemler hakkındaki bilgiler kullanılarak mevcut bir işlem modeli genişletilir veya iyileştirilir [1].



Şekil 1. Süreç madenciliğinin üç ana bileşeni: keşif, uygunluk ve iyileştirme

Süreç madenciliği teknikleri kullanılarak farklı alanlarda yapılan çalışmalar mevcuttur. Van der Aalst ve arkadaşları tarafından yapılan çalışmada [2], süreç madenciliği teknikleri kullanılarak, taşeronlar ve tedarikçiler tarafından gönderilen faturaların işlenmesi üç farklı bakış açısıyla analiz edilmiştir: (1) süreç perspektifi, (2) organizasyonel bakış açısı ve (3) vaka perspektifi. Bu amaçla, süreç madenciliğinde yaygın olarak kullanılan ProM uygulaması ile, genel olarak süreç madenciliğinin kendi vakalarındaki uygulanabilirliğini ve kullandıkları algoritma ve araçları ortaya koymuşlardır.

Poncin ve arkadaşları tarafından yapılan diğer bir çalışmada [3], sürüm kontrol sistemleri, yazılım hata kaydı sistemleri gibi yazılım depolarında tutulan veriler kullanılarak süreç madenciliği tekniklerinin yazılım geliştirici faaliyetleri üzerine uygulanması önerilmiştir. Süreç madenciliğinin uygulanabilir olması için, farklı depoları eşzamanlı olarak analiz etme ve elde edilen bilgileri birleştirme gibi zorunluluklar ortaya çıkmıştır. Çalışma ile çeşitli vakalara ait yazılım hata kaydı yaşam döngüsü ortaya konulmuştur.

Gadler ve arkadaşları tarafından yapılan çalışmada [4], bir yazılım sistemiyle etkileşime giren kullanıcıların işlemlerini gösteren süreç modellerinin otomatik olarak nasıl oluşturulabileceği incelenmiştir. Bu amaçla Saklı Markov modeli kullanılarak başarılı bir şekilde süreç modeli ortaya konmuştur.

Çalışmamızda seçilen bankacılık uygulaması ile ilgili ekranlardan yapılan her işlemde, arka planda çağrılan servislere ait işlem günlükleri kullanılarak süreç akışının otomatik olarak çıkarılması hedeflenmiştir. Bildirinin organizasyonu şu şekildedir: Bölüm 2'de Endüstriyel vaka ve problem tanımı anlatılmış, Bölüm 3'te yöntem sunulmuş, Bölüm 4'te çalışma sonuçları değerlendirilmiş, son olarak Bölüm 5'te çalışma özetlenmiş ve planlanan gelecek çalışmalar belirtilmiştir.

2 Endüstriyel Vaka ve Problem Tanımı

İlgilendiğimiz süreç bankacılıkta ticari kredi başvurularının yapıldığı ekran ve uygulamaları içermektedir. Ticari kredi başvurusunda bulunan müşteriler şube tarafından değerlendirildikten sonra uygun ürün, limit, vade ve diğer özellikler seçilerek ekranlardan girişleri yapılmaktadır. Girilen kriterlere göre müşteriye bir fiyat verilmekte ve verilen fiyata göre ödeme planı oluşturularak ve müşteri onayı alınarak başvurusu alınmaktadır. Başvuru bundan sonra tahsis modülüne iletilerek kredi verilip verilmeyeceği karar motoru vasıtasıyla sonuçlandırılmaktadır. İlgilendiğimiz süreç tahsis modülüne iletilene kadar olan başvuru sürecini içermektedir.

Günde toplam olarak yaklaşık 3,000 adet başvuru alınmakta, şube çalışanlarınca birbirini takip eden 3 farklı ekrandan girişler yapılmaktadır. Uygulama çoğunlukla 09:00-17:00 saatleri arasında kullanılmakta ve bu saatler arasında ayakta olması beklenmektedir. Sistem, Windows sunucuları üzerinde çalışan .net wcf (windows communication framework) uygulamalarından oluşmaktadır.

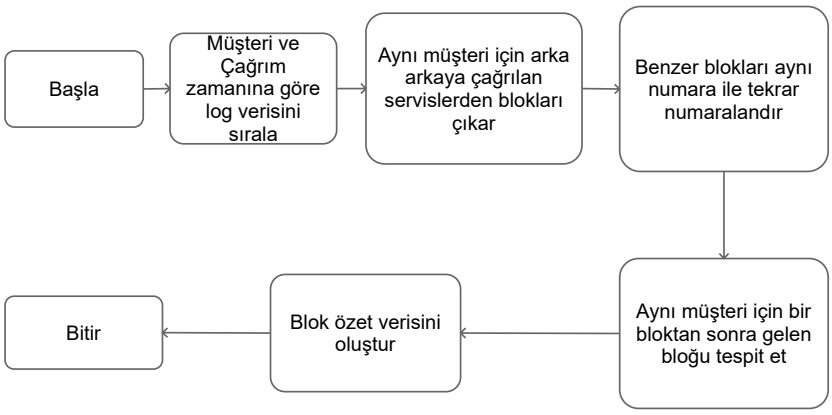
2.1 Problem Tanımı

Sistemde yazılımsal bir sorun oluşması durumunda müşteri şikayetine neden olunmaması adına çok kısa sürelerde sorunun tespit edilmesi ve çözümlenmesi beklenmektedir. Soruna neden olan servisin hızlı bir şekilde tespit edilebilmesi için güncel süreç akışının ortaya konulmuş olması gerekmektedir. Ancak yazılım güncellemeleri ve yeni süreçler eklenmesi ile sistem sürekli bir şekilde değişmekte ve daha önce ortaya konulmuş olan statik modeller kısa sürede güncelliğini yitirebilmektedir. Bu çalışmadaki amacımız log verisini kullanarak otomatik olarak süreç modelini ortaya koyabilmektir.

3 Yöntem

Bu bölümde işlem günlüğü verisi kullanılarak sistemi modellemek için ortaya koyduğumuz yöntem anlatılmaktadır. Yöntemimiz ile ilgili genel yaklaşım Şekil 2'de verilmiş olup detayları devam eden alt bölümlerde verilmiştir.

Ticari kredi başvuru sistemi arayüzü ile gerçekleştirilen aksiyonların herbiri **işlem** olarak tanımlanmıştır. Verilerin elde edildiği Ticari kredi başvuru sisteminde tanımlanmış olan iş akışını tanımlayan görevlere **süreç** denmektedir. Süreçlerin gerçekleştirilmesi için koşulan kod parçaları **web servisi** olarak tanımlanabilir. Bir süreci oluşturan birbiri ile ilişkili olan ve arka arkaya koşturulan web servislerine ise **blok** denmektedir.



Şekil 2. Log verisinden işlem bloklarının otomatik olarak çıkarılmasına dair yöntemin özet akışı.

3.1 İşlem Günlüğü Kayıtlarının Oluşturulması

Web servisleri çağrıldığında her bir servis ilgili veritabanına, çağrılma zamanı, servis adı, cevap verme süresi, servisin girdi ve çıktı alanlarını içeren işlem günlüğü kaydını oluşturmaktadır. Servisin girdi ve çıktısındaki bilgiler kullanılarak ilgili çağrıda geçen müşteri numarası, çağrı sonucunda iş hatası veya sistem hatası alınıp alınmadığı hazırlanan program ile otomatik çıkarılmıştır. Veritabanında her bir servis çağrımında tutulan kayıtlara ait bilgiler Tablo 1'de verilmiştir.

Tablo 1. İşlem günlüğü tablosunda tutulan bilgiler.

Alan adı	Alanın açıklaması
Müşteri numarası	Servisin hangi müşteri için çağrıldığı
Çağrım zamanı	Servis çağrısı başlangıç anı
Yanıt süresi	Milisaniye cinsinden servisin yanıt verme süresi
Çağrılan servisin adı	Sistemde tanımlı olan servis adı
İş hatası	Servis çağrısı sonucunda iş hatası alınıp alınmadığı
Sistem hatası	Servis çağrısı sonucunda sistem hatası alınıp alınmadığı

3.2 İşlem Bloklarının Çıkarılması

Kullanıcılar ekrandan tek bir işlem yaptıklarında arka tarafta birden çok servis çağrılabilmektedir. Ekrandan yapılan tek bir işlemde arka arkaya çağrılan servisler aynı işlem bloğu içinde değerlendirilmiştir. Bir müşteri için yapılan bir işleme ait servis çağrılarını çıkarabilmek için öncelikle işlem günlüğü kayıtları müşteri numarası ve çağrım zamanına göre sıralı bir şekilde oluşturulmuştur.

Servislerin arka arkaya çağrıldığı sonucuna ulaşabilmek için, o müşteri için çağrım zamanı ve yanıt süresi alanındaki veri kullanılmıştır. Bir müşteri için bir servis çağrıldıktan ve yanıt alındıktan sonra aynı müşteri için 1 saniye içinde sıradaki servis çağrılmışsa aynı işlem bloğu içinde değerlendirilmiştir.

3.3 Benzer İşlem Bloklarının Tespiti

İşlem blokları çıkarıldıktan sonra benzer bloklar tespit edilmiştir. Ekrandan aynı işlem farklı müşteriler için yapıldığında arka tarafta çağrılan servisler birebir aynı olmayabilmektedir. Bunun birinci sebebi servis çağrım çıktıları ön belleğe (cache) alınabilmekte, ön belleğe alınan veri ekrandaki ilgili işlem tekrar yapıldığında, ön bellek süresince kullanılabilir durumdaysa servis çağrımlarından ilgili olanlar yapılmamaktadır. İkinci nedense, birbirinin çıktılarını kullanmayan servisler olabildiği için servislerin çağrımları asenkron yapılabilmektedir, dolayısıyla sıralama da değişebilmektedir.

Bu nedenlerden dolayı, iki blokta çağrılan servisler, sıraya bakılmaksızın % 70 oranında tutuyorsa bloklar aynı olarak değerlendirilmiş ve buna göre numaralandırılmıştır.

3.4 İşlem Blokları Arası Geçişler

Bloklar tespit edildikten sonra aynı müşteri için ekrandan arka arkaya yapılan işlemlerin (blokların) neler olduğu belirlenmiştir. Bir bloğa ait başlangıç ve bitiş zaman bilgisi elimizde bulunmaktadır. Kullanıcılar ekrandan bir müşteri için işlem yaptıklarında, belli bir süre içinde, tamamlamaları gereken diğer işlemi yapmaktadırlar. Ancak ekran açıkken işlemin hemen tamamlanması gerekmemekte, kullanıcılar aralar verebilmektedirler. Bu nedenle 20 dakika kadar bir süre içinde aynı müşteri için yapılmış olan işlem bloklarının arka arkaya geldikleri varsayılmıştır. Buna ek olarak, kullanıcılar işlemleri yarım bıraktıktan sonra tekrar başa

dönebilmektedir-ler. Ancak başlangıç işlemi sabit olduğundan diğer bloklardan başlangıç işlemine geçiş tespit edilirse sürecin yeniden başladığı varsayılmıştır.

Sonuç olarak, işlem günlüğü verisinden yola çıkılarak işlem bloğu özet çıktısı oluşturulmuştur. Bu çıktı, servis adı, ilgili blok numarası, sonra çağrılan blok numaraları alanlarını içermektedir. Oluşturulan veriye ait alanlar ve açıklamaları Tablo 2'de verilmiştir:

Tablo 2. Sonuç olarak oluşturulan işlem bloğu özet tablosunda tutulan bilgiler.

Alan adı	Alanın açıklaması
Servis adı	Sistemde tanımlı olan servis adı
İlgili blok	Servisin ait olduğu işlem bloğuna verilen numara
Bu bloktan sonraki işlem bloğu	İlgili işlem bloğundan sonra gelen olası işlem blokları

Algoritma sonucunda bloklara ait numaralar elde edilmiş, alan bilgisine sahip yazılımcı tarafından numaralandırılmış olan işlem blokları yaptıkları işe göre isimlendirilmiştir.

3.5 Süreç Modelinin Ortaya Konulması

Elde edilen veriden süreç modelinin görselleştirilmesi için Graphviz isimli bir uygulama kullanılmıştır [5]. Graphviz açık kaynak bir grafik görselleştirme uygulamasıdır. Ağ oluşturma, biyoinformatik, yazılım mühendisliği, veritabanı ve web tasarımı, makine öğrenmesi ve diğer teknik alanlarda görselleştirme amacıyla kullanılmaktadır. Servis adı, ait olduğu blok ve ilgili bloktan sonra çağrılan blok bilgilerini içeren veri Graphviz uygulamasının girdi olarak aldığı formata çevrilmiştir.

4 Vaka Çalışması Sonuçları

Bu bölümde yapılan çalışmaya ait çıktı örnekleri ve ortaya çıkan model sunulmuştur. Endüstriyel vakadan 2 gün boyunca 800,000 adet işlem günlüğü kaydı elde edilmiştir. Örnek kayıtlarda web servisin girdisinde geçen orijinal müşteri numaraları şifrelenerek ilgili müşteri numaraları elde edilmiştir. Web servisin çıktısından da iş hatası veya sistem hatası alınıp alınmadığı ortaya konmuştur. 0 değeri hata alınmadığını, 1 değeri ise hata alındığını ifade etmektedir. Elde ettiğimiz işlem günlüğü verisine ait özet bilgi Tablo 3'te, örnek bazı kayıtlarsa Tablo 4'te verilmiştir.

Tablo 3. Elde edilen veriye ait özet bilgi.

Kayıt adedi	Zaman aralığı	Farklı web servis adedi	Dosya boyutu
800.000	07.05.2019 - 08.05.2019	196	55,6 MB

Tablo 4. Örnek işlem günlüğü kayıtları.

Müşteri no.	Çağrım zamanı	Yanıt süresi (ms)	Çağrılan servisin adı	İş hatası	Sistem hatası
0	07.05.2019 08:34:50.477	46	ELGBT:retrieveBasicInfo	0	0
0	07.05.2019 08:34:50.513	93	ELGBT:retrieveCustomerGroupsAndFirms	0	0
0	07.05.2019 08:34:57.843	31	AppMngUI:IsEligible	0	0
0	07.05.2019 08:34:57.873	62	ELGBT:findMessage	0	0
0	07.05.2019 08:35:43.100	15	AppMngUI:RetrieveDBParameters	0	0

Bölüm 3'te anlatılan yöntemle işlem günlüğü kayıtları gruplanarak işlem blokları elde edilmiş, bu bloklardan birbirine benzeyenler aynı blok olarak numaralandırılmıştır. Tablo 5'te örnek iki işlem bloğu verilmiş, baz alınan blokta geçen 6 adet servisten 5'i diğer blokta da geçtiği, dolayısıyla % 83 oranında benzediği için (5/6) aynı blok olarak değerlendirilmiştir.

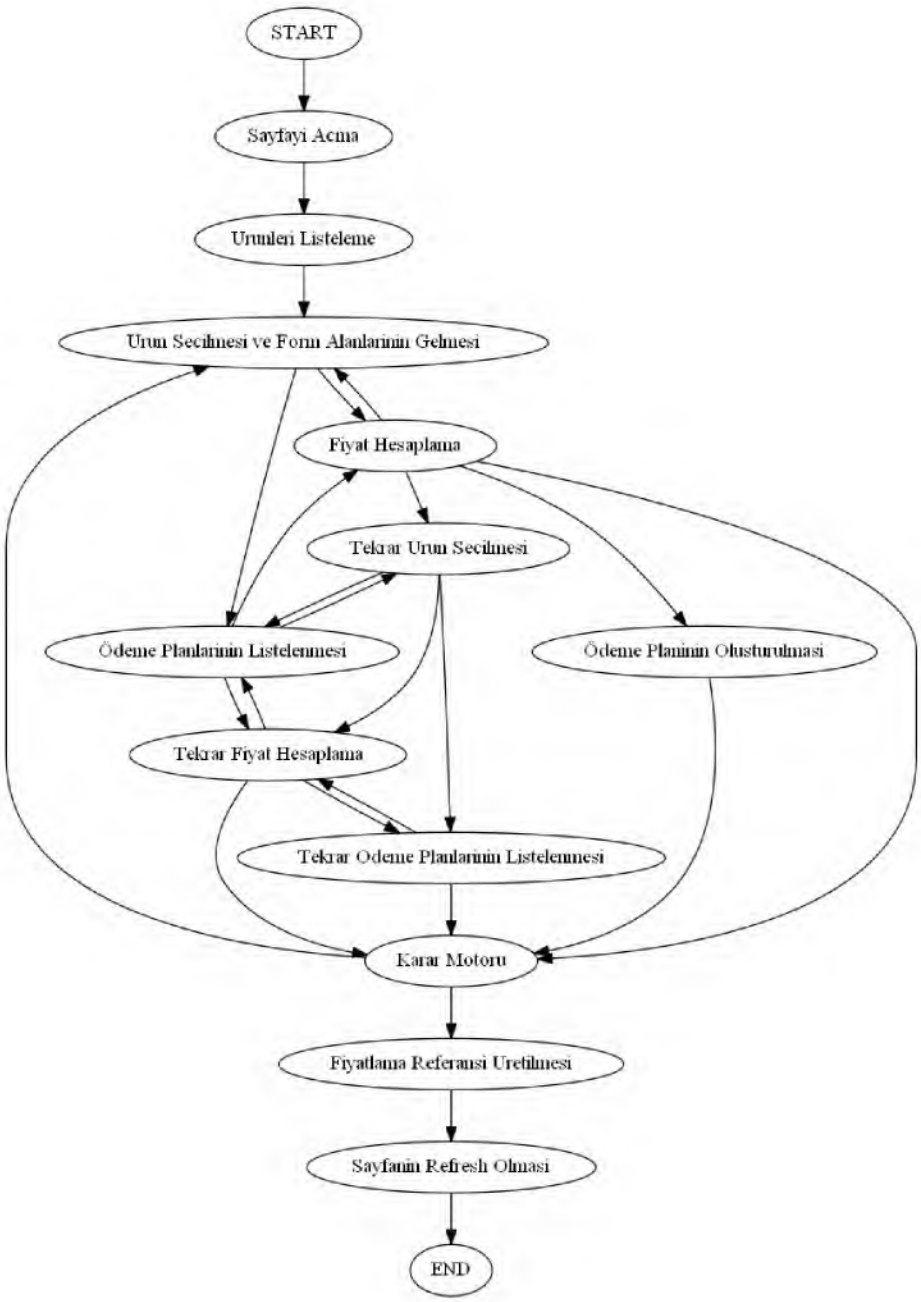
Tablo 5. Benzer iki işlem bloğu.

Baz alınan blokta geçen servisler	Benzer olarak değerlendirilen blokta geçen servisler
AppMngUI:retrieveFinancialInfo	AppMngUI:retrieveFinancialInfo
AppMngUI:retrieveScore	AppMngUI:retrieveCustomerByReferenceId
AppMngUI:retrieveCustomerByReferenceId	AppMngUI:retrieveScore
AppMngUI:calculateProfileRate	AppMngUI:calculateProfileRate
AppMngUI:retrieveProductParameterList	AppMngUI:calculateCorporateProductProposedRate
AppMngUI:calculateCorporateProductProposedRate	

Sonuç olarak ilgili servis, ait olduğu işlem bloğu ve bu bloktan sonra gelebilen olası işlem blokları ortaya konulmuş, örnek bazı kayıtlar Tablo 6'da verilmiştir. 800.000 adet işlem günlüğü kaydından 12 adet işlem bloğu çıkarılmıştır. Bu bloklar alan bilgisine sahip yazılımcı tarafından isimlendirilmiş, elde edilen süreç akışı ise Şekil 3'te verilmiştir. Çıkarılan süreç akışı ekibimiz dışından süreç bilgisine sahip analist tarafından da incelenerek işlem bloklarının ve bu bloklar arasındaki geçişlerin doğru olduğu teyit edilmiştir.

Tablo 6. İşlem bloğu tablosu örnek kayıtları.

Servis adı	İlgili blok	Bu bloktan sonraki işlem bloğu
AppMngUI:retrieveMainProductValueList	2	3
AppMngUI:retrieveReferenceDataList	2	3
AppMngUI:retrieveProductBriefInfoList	3	4,8,9
AppMngUI:RetrieveDBParameters	3	4,8,9
AppMngUI:calculateProfileRate	3	4,8,9



Şekil 3. Ticari kredi başvurusu için log kayıtlarından çıkarılan süreç akışı.

5 Sonuç ve Gelecek Çalışmalar

Bu çalışmayla bir bankacılık uygulamasına ait işlem günlüklerinden işlem bloklarını ve bu bloklar arası geçişleri çıkardık ve süreç modelini ortaya koyduk. Önceki çalışmamızda [6] web servislerinin belli aralıklarda ortalama yanıt dönme süreleri ve kaç kez çağrıldıkları bilgilerini kullanarak aralarındaki bağımlılıkları ortaya çıkarabileceğimizi görmüştük. Önümüzdeki dönemde ise hem web servis hem de işlem bloğu bazlı olarak sistemde bir anomalite olup olmadığına dair tahminleme yapılması için çalışmalarımıza devam edeceğiz. Bu sayede kullanıcıların ekranlarda yapamadığı veya sayısında düşüş olan bir işlem bloğu olduğunda otomatik olarak uyarı verebileceğimizi öngörmekteyiz.

Kaynakça

1. Van Der Aalst, W., Adriansyah, A., De Medeiros, A. K. A., Arcieri, F., Baier, T., Blickle, T., ..., Burattin, A. (2011, August). Process mining manifesto. In International Conference on Business Process Management (pp. 169-194). Springer, Berlin, Heidelberg.
2. Van der Aalst, W. M., Reijers, H. A., Weijters, A. J., van Dongen, B. F., De Medeiros, A. A., Song, M., Verbeek, H. M. W. (2007). Business process mining: An industrial application. Information Systems, 32(5), 713-732.
3. Poncin, W., Serebrenik, A., Van Den Brand, M. (2011, March). Process mining software repositories. In 2011 15th European Conference on Software Maintenance and Reengineering (pp. 5-14). IEEE.
4. Gadler, D., Mairegger, M., Janes, A., Russo, B. (2017). Mining Logs to Model the Use of a System. International Symposium on Empirical Software Engineering and Measurement, 2017, 334–343.
5. Ellson, J., Gansner, E., Koutsofios, L., North, S. C., Woodhull, G. (2001). Graphviz—open source graph drawing tools. In International Symposium on Graph Drawing (pp. 483-484). Springer, Berlin, Heidelberg.
6. Aktaş, E. U., Çalpur, M. Ç., Yıldırım, Ü. Ü., Yıldırım, E. (2018) Inferring Dependencies Among Web Services with Predictive and Statistical Analysis of System Logs. Turkish National Software Architecture Conference.

Yazılım Geliştirme Dış Kaynak Alımlarında Startup Firmalar İçin Yeni Bir Satın Alma Modeli

Selin Çalışkan[1], Özlem Özenç[1], Şerife Çağla Özgöç[2], Okan Horuz[2]

[1,2] Turkcell İletişim Hizmetleri A.Ş, İstanbul, Türkiye
{caliskan.selin, ozlem.yaman, serife.ozgoc, okan.horuz}
@turkcell.com.tr

Özet. Bu bildiride, Türkiye'de dijital telekomünikasyon şirketi olarak faaliyet göstermekte olan ve aynı zamanda "yerlilik ve millilik" kavramını tüm fonksiyonlarında ön planda tutan bir şirketin, teknoloji geliştirme alanında çalışmalar yapan Startup'lara somut faydalar sağlayabilmek için kurguladığı bir yazılım geliştirme dış kaynak satın alma modelinden bahsedilecektir. Bu modelin şirketin alım süreçlerinde ne gibi farklılıklar yarattığına ve bu süreçlerde rol oynayan tüm paydaşların iş yapış şekillerindeki Startup odaklılık dönüşümüne değinilecektir.

Anahtar Kelimeler: Yazılım Dış Kaynak Satın Alma, Girişimci Firmalar, Startup Ekosistemi, Startup İş Birliği Modeli

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

A New Buying Model For Startup Companies At Software Development Outsourcing Procurement

Selin Çalışkan[1], Özlem Özenç[1], Şerife Çağla Özgöç[2], Okan Horuz[2]

[1,2] Turkcell İletişim Hizmetleri A.Ş, İstanbul, Türkiye
{caliskan.selin, ozlem.yaman, serife.ozgoc, okan.horuz}
@turkcell.com.tr

Abstract. In this paper, A digital telecommunication company which is native and national player in Turkey that sets a new software development outsourcing procurement model up to provide Startups working in the field of technology development will be mentioned. The differences that this model creates in the company's procurement processes and the transformation of the Startup focus of all stakeholders involved in the procurement processes will be discussed.

Keywords: Software Outsourcing Procurement, Entrepreneurship, Startup Ecosystem, Startup Collaboration Model

1 Giriş

Hızla dijitalleşen dünyada, bilim ve teknolojideki inovasyonların ortaya çıkma ve yayılmalarındaki artışın yanı sıra, yeni teknolojilerin multidisipliner ve kompleks nitelikleri yüzünden, şirketlerin değişen teknolojik süreçlere kendilerini adapte edebilme güçleri azaldı. Henüz olgunlaşma süreçlerini tamamlamadan bir diğeri konuşulmaya başlanan yeni teknolojiler, şirketlerin AR-GE faaliyetlerini ve dolayısıyla ülkelerin AR-GE'ye karşı yaklaşımlarını ve kurumların oluşturduğu AR-GE ekosistemlerini değiştirdi.[1] Bu teknolojik gelişmelerin hızı ve yoğunluğu, AR-GE faaliyetlerinde daha dinamik bir ekosistem ihtiyacı doğurdu. Bu noktada, değişime hızla adapte olabilen, daha çevik, belirli alanlara odaklanmış olarak çalışan, daha yenilikçi teknoloji Startup'larının sayısı ve etkinliği giderek arttı. Startup kavramı, Türkiye'de hala yeni bir kavram olmakla birlikte, ekosistemin daha somut şekilde desteklenme ihtiyacı devam etmektedir.

Pek çok Startup'ın çözülemeyen bir problem olarak gördüğü "ticarileşme" sorunsalı, fikirlerinin işletilmesi ve gerçek dünyada değer üretmeye başlaması ile ticari bir karşılık bulabilmektedir. Buna aracılık sağlayabilecek ve Startup'ların pazara erişebilmelerinde rol oynayacak en önemli aktör

olarak kurumsal şirketler ön plana çıkmaktadır. Finansmanın dışında, Startup'lar kendilerini yönlendirebilecek ve gerektiğinde eğitebilecek teknik kapasiteye sahip, kendilerine etkili bir profesyonel ilişki ağı sağlayabilecek ve küreselleşme potansiyellerini besleyebilecek kanallara da ihtiyaç duymaktadırlar. Ticarileşme süreçlerini çok önceden tamamlamış kurumsal şirketler, bu konular özelinde bir destek mekanizması oluşturmak ve Startup'ları bir adım daha ileriye taşıyacak iş birliği modelleri kurgulamak konusunda yetkin durumlardırlar.

Diğer yönden bakıldığında, kurumsal şirketlerin teknolojiye hızlı adapte olabilmek, teknolojinin çıktılarını müşteriye katma değerli bir şekilde sunabilmek ve bu sayede yeni müşterilere erişebilmek ve en önemlisi rekabetçi pazarlarda mevcut varlıklarını koruyabilmek gibi sebepler, Startup'larla iş birlikteliklerinin önünü açmaktadır. Küresel açıdan bakıldığında, pek çok uluslararası şirketin Startup'larla iş birliğine gittiği ve bu şirketlerin diğerlerine oranla daha hızlı bir ivmeyle büyüdüğü gözlemlenmektedir. Bu küresel çaplı deneyimlerle birlikte, şirketler ve Startup'lar arasındaki ilişki her geçen gün daha çok yatırıma değer görülen bir ortaklık haline gelmektedir. Zira karşılıklı yarar sağlamaya ve hızlı büyümeye katkı sağlayabilecek bu birleşmeler, her iki tarafa da fayda sağlamanın yanı sıra güçlü iş ortaklıklarına ve birliktelikten doğacak yeni inovasyonlara imkân sağlayabilmektedir.[2]

Bu iş birlikteliklerinin başarılı şekilde yürütülmesi paydaşların doğaları gereği oldukça zordur. Kurumsal şirketlerin prosedürlerine ve süreç akışlarına daha sadık olması, bu şirketleri Startup'lardan zaman algısı açısından ayırmaktadır. Her iki paydaşın aynı iletişim kanallarını kullanamaması da birlikte çalışmalarını zorlaştıran unsurlardandır. Bu yüzden kurumsal şirketlerin Startup'larla çalışması, Türkiye'de henüz riskli bir iş birlikteliği modeli olarak görülmektedir ve riski alamayan şirketler neticesinde sayıca yetersiz kalmaktadır.

Bu bildiride, Türkiye'de dijital telekomünikasyon şirketi olarak faaliyet göstermekte olan ve aynı zamanda "yerlilik ve millilik" kavramını, tüm fonksiyonlarında ön planda tutan bir şirketin, teknoloji geliştirme alanında çalışmalar yapan Startup'lara somut faydalar sağlayabilmek için geliştirdiği bir yazılım geliştirme dış kaynak satın alma modeline değinilecektir. Bu modelin şirketin alım süreçlerinde yarattığı farklılaşmadan ve bu süreçlerde rol oynayan paydaşların iş yapış şekillerindeki Startup odaklılık dönüşümü aktarılacaktır. Son olarak modelin uygulanması sırasında karşılaşılan temel sorunlardan ve bu sorunların çözümü için geliştirilen çözümlerden bahsedilecektir.

2 Yazılım Geliştirme Dış Kaynak Satın Alma Modeli

Türkiye'de dijital telekomünikasyon şirketi olarak faaliyet göstermekte olan şirketin, yazılım geliştirme birimleri pek çok farklı alanda çalışmalar yapmaktadır. Fakat şirketin hem iç müşteri sayısının fazlalığı, hem de Türkiye'deki Pazar liderliğini elinde bulundurması sonucu dış müşteri sayısının fazlalığından kaynaklı olarak, yazılım geliştirme ihtiyaçlarının tamamını kendi iç kaynakları ile yapması mümkün değildir. Bu yüzden şirket yazılım geliştirme faaliyetlerinin önemli bir kısmını dış kaynakların desteği ile yürütmektedir. Bu yazılım geliştirme alımları, kozmik işlev puanlandırma (CFP) metriği ile yapmaktadır. Geliştirilecek yazılım için yapılacak alım, adam/gün ekseninde değil, yazılımın kapsam boyutunun ihale edilmesi şeklinde yapılmaktadır.[3]

2.1 Startup İş Birlikteliği Modeli Kurgusu

Şirketin yazılım dış kaynak hizmeti almak için ihale ettiği işler kapsam olarak büyük ve yapısal olarak karmaşık işlerden oluşmaktadır. Bu bağlamda, bu işlerin yalnızca tek bir Startup firma ile yürütülmesi, Startup'ların kaynak havuzları ve yönetsel tecrübelerinin yetersizliği yüzünden verimli kabul edilmemektedir. Bunun yanında yazılım birimleri, yalnızca Startup'lardan oluşan dış kaynak alımlarına karşı, işin sürekliliğini ve kalitesini tehlikeye atabileceği riski barındırdığı gerekçesi ile direnç gösterebilmektedir.

Bu çekinceler sonucunda, yazılım geliştirme dış kaynak satın alma birimi önceden tamamının ihale edildiği işlerin kapsamını %90 ve %10 olarak ayırmıştır. Kapsamın %90'ı ticarileşme sürecini tamamlamış ve kurumsal firmalarla çalışma tecrübesi olan Ana Firma'lara ihale edilirken, %10'luk kısmı "ihale edilmeksizin", şirketin oluşturduğu Startup havuzundan teknik yetkinlikleri sağlayacak şekilde seçilmektedir. Yapılan teknopark ziyaretleri ve anketler sonucu, firmalar 5 ana kriteri sağlamaları sonucunda havuza dahil edilmiştir. Öncelikli olarak bir Teknopark bünyesinde yer almaları, geliştirmekte oldukları bir AR-GE projelerinin olması, firma yaşının 0-5 yaş aralığında olması, çalışan sayısının 15 ve altı olması veson olarak, yıllık cirolarının 1 Milyon Liranın altında olması kriterlerini sağlayan firmalar, ihale özelinde teknik açıdan değerlendirmeye tabi tutulmaktadır. En yüksek puanlandırmayı alan Startup'a, Ana Firma ile anlaşılan CFP birim bedelinin en çok %15 fazlası olacak şekilde teklif sunulmaktadır. Startup firmanın teklife uygunluğunu iletmesi sonucu, üç paydaştan (Şirket, Ana Firma, Startup) oluşan bir çalışma ekibi oluşturulmaktadır. Bu modelde

“Startup”, hem “Şirket” hem de “Ana Firma” ile birlikte çalışmaktadır. Startup yönetsel her türlü konuda ve sağladığı servis seviyesi hedeflerinin kontrolü konusunda Ana Firma’ya bağlıdır. Fakat ödemelerini doğrudan Şirket tarafından alacağı için direkt olarak Şirket’in bir tedarikçisi konumunda olacaktır. Ana Firma’dan beklenen, Startup’a mentörlük sağlaması ve Şirket kontrolünde bir çıraklık programı kurulmasıdır. Bu sayede Startup, şirketin direkt tedarikçisi olmanın verdiği referansla iş ağlarını genişletebilecek ve Ana Firma ve Şirket’le birlikte çalışmasının sonucunda kurumsal süreçler ve teknik konularda derinlemesine bilgi ve tecrübe elde edebilecektir. Ek olarak, aylık ürettiği CFP kapsamı doğrultusunda ödeme alacak olan Startup, kırılgan nakit akışlarını düzenli olarak besleyebilecektir.

2.2 Startup Odaklı Satın Alma Modeli Adaptasyonu

Yazılım geliştirme dış kaynak satın alması pek çok birime dokunmaktadır. Bu yüzden alım kararındaki bu değişiklik için tüm paydaşlarla toplantı yapılmış ve süreçler tekrar tasarlanmıştır. Modelin entegrasyonu sırasında, tüm paydaşların sürekli etkileşimleri ve geri bildirimleri sayesinde, iyileştirme alanları anlık olarak takip edilmiş ve modelin optimizasyonu için gerekli iyileştirmeler yapılmıştır.

2.1.1 Teknoloji Birimleri

Var olan alım modelinde yapılması düşünülen değişiklik için şirketin teknoloji birimlerinde çalışan müdürlerle görüşülerek fikir alışverişi yapılmış ve modelin kurgusunda iyileştirmeler yapılmıştır. Bu toplantılarda, şirketin hangi kurumları Startup olarak değerlendireceği netleştirilmiş ve Startup’ların teknik olarak standart bir değerlendirmeye tutulmaları konusunda ortak karar alınmıştır.

2.1.2 Teknoparklar

Geçiş sürecinde Türkiye’de yer alan tüm teknoparklara çeşitli anketler yollanmış ve genel başvurular toplanmıştır. Uzaktan erişimin dışında, başta İstanbul’daki teknoparklar olmak üzere, Ankara’da ve çeşitli illerde yer alan teknoparkların yönetimleri ziyaret edilmiş ve iş birlikteliği modeli hakkında görüşler alınmıştır. Hem anketler, hem de yapılan ziyaretler sonucu gelen başvurular, şirketin belirlediği Startup olma koşulunu sağlaması doğrultusunda birebirde tekrar görüşülerek şirketin Startup havuzuna eklenmiştir.

2.1.3 Denetim

Satın alma süreçlerini denetleyen ekiplerin görüşlerine başvurularak, şirketin iç dinamikleri ile çatışmayacak bir model olduğu konusunda mutabakata varılmıştır.

2.1.4 Teknoloji Süreç Geliştirme Ekibi

Teknoloji birimlerinin iş süreçleri ile ilgilenen birimlerle birlikte, alımı yapılacak hizmetlerin teknik şartname taslakları, Startup modelini kapsayacak şekilde tekrar düzenlenmiş ve ilgili tüm birimlere gönderilmiştir. Bu sayede iş birlikteliği modeli, ilgili tüm yazılım birimlere resmi olarak duyurulmuştur. Bunun dışında satın alma birimlerinin kullandığı sipariş ekranları ve sözleşme ekranlarına Startup seçeneği eklenmiştir. Bu sayede Şirket, Startup'lar için de aylık ve yıllık raporlara ulaşabilecek ve Startup hedeflerinin konumlandırılmasını daha sağlıklı yapabilecektir.

2.1.5 Hukuk

Şirket ve Ana Firma'dan oluşan çift taraflı sözleşmeler, Startup firmaların haklarını da içerecek ve koruyacak şekilde, üç taraflı sözleşmeler olarak tekrar düzenlenmiştir.

3 Sonuç

Mayıs 2019 itibariyle aktif bir şekilde ihalelerde kullanılmaya başlanan modelin Startup'lar için somut kazanımı, yalnızca bir ayda 4 farklı teknoparkta yer alan, 5 farklı Startup firma ile doğrudan iş birlikteliği ve bu firmalardan yapılacak, senelik 1,37 Milyon Türk Lirası dış kaynak hizmet anlaşması olmuştur. [Fig.1] Bu model değişikliği ile birlikte, gelen tüm yazılım geliştirme dış kaynak talepleri, Startup odaklı olarak değerlendirilmektedir ve ihale yönetimleri bu şekilde yapılmaktadır.



Şekil 1. Dönüşümün İlk Ayında Yapılan İş Birliktelikleri – Mayıs 2019

Modelin uygulandığı ilk 5 ihalede karşılaşılan temel sorun, sözleşme aşamasında mutabakat sürecinin uzaması olarak gözlemlenmiştir. İhale sürecinde firmalarla yapılan teknik görüşmeler esnasında modelin detayları, hem sözel hem de şartname ile aktarılmış olmasına rağmen, hizmet sözleşmesinin imzalanması aşamasında Ana Firma'larla mutabık kalmak zorlayıcı olmuştur. Bu sorunu ortadan kaldırmak adına, ihale öncesinde şartnamelerin imzalı olarak, ilgili satın almacıya gönderilmesi şeklinde bir değişiklik yapılmıştır. Firmaların şartname içinde yer alan Startup modelini içeren maddelere gösterdikleri uyumsuzluklar, ihale sürecinde teknik notlandırmalarına direkt olarak olumsuz etki edecek şekilde dikkate alınmaya başlanmıştır.

Modelin sürekliliği ve şeffaflığı açısında, tek taraflı yarardan ziyade kazan-kazan ilişkisi kurulması amaçlanmaktadır. Bu noktada yalnızca Startup, şirkete ne sağlar sorusu ile değil, Startup'a sunulan değer teklifi nedir, bu işbirliği Startup'ın büyümesine katkı sağlar mı, Startup'ın hızla büyümesi için finansman dışında hangi şirket kaynakları açılabilir, böyle bir iş birliği Startup'ı yavaşlatır mı gibi pek çok sorunun üzerinde durulmaktadır. Cevaplar doğrultusunda model üzerinde iyileştirmeler ve eklemeler her paydaşın yorumu alınarak düzenli olarak yapılmaktadır.

4 Teşekkür

Modelin hayata geçirilmesi ve tanıtılması için katkılarından dolayı Tedarik Zinciri'nden Sorumlu Genel Müdür Yardımcımız Ali Türk'e, ihale yönetimindeki değişiklik sürecini takibinden ve yönlendirmelerinden dolayı Teknik Dışı Satın Alma Direktörümüz Attila Yunusoğlu'na, Startup havuzu oluşturma konusunda desteklerinden dolayı Tedarik Zinciri Süreç Geliştirme Ekibine ve bu model uygulanarak yapılan ilk pilot ihalemizdeki iş birliğinden dolayı İş Uygulamaları ve Öğrenme Çözümleri Müdürü Selami Bağrıyanık'a teşekkürler.

5 Kaynakça

1. Arslanhan, S: Yeni Teknolojiler ve Startuplar Türkiye Ekonomisi İçin Fırsattır, http://www.cleanroomnews.org/yeniteknolojiler-ve-Startuplar-turkiye-ekonomisi-icin-firsattir, erişim tarihi 2018/03/20.

2. T.C. Enerji ve Tabii Kaynaklar Bakanlığı: Startuplar Yaklaşım Belgesi, https://www.enerji.gov.tr/file/?path=root%2f1%2fdocuments%2fsayfalar%2f20190508-20191b.pdf, erişim tarihi 2019/05/08.

3. Deveci, A., Çalışkan, S., Bağrıyanık, S.: Yazılım Geliştirme Hizmetlerinin Satın Alınması için Bir Maliyet Modeli Önerisi. Turkish National Software Engineering Symposium (2018).

Yazılım Yaşam Döngüsü Modelleri için Sistematik Eşleme Çalışması

Yagup Macit[1], Ahsen İkbal Aytekin[1], Çağdaş Üsfekes[1]

[1]HAVELSAN Askeri Yazılımlar Grup Müdürlüğü
06510 Ankara, Türkiye
{ymacit, ayergok, cusfekes}@havelsan.com.tr

Özet. Uygulama yazılımlarının fikir aşamasından bakım aşamasına kadar olan tüm aşamalarını kapsayan etkinlikler bütünlüğü, Uygulama Yaşam Döngüsü Yönetimi (UYY) süreci olarak tanımlanmaktadır. Sürecin içerisinde bulunan yöntem ve pratikler güncel ihtiyaçlar ve olanaklara göre evrimsel olarak değişkenlik göstermektedir. Yazılım geliştirme yöntem ve pratiklerinin evrimi, endüstri ve akademik çevrelerin işbirliği ve rekabeti altında süregelmektedir. Evrimsel olarak ortaya çıkan her bir yöntem ve pratiğin yaşam eğrisi, kapsama alanı ve soyutlanma düzeyine göre faklılıklar gösterebilmektedir. Yine her bir yöntem ve pratiğin tüm yöntem ve pratikler içerisindeki ağırlığı ve baskınlığı değişebilmektedir. Ayrıca endüstriyel yazılım geliştiriciler kendi yöntem ve pratik ihtiyaçları için akademik yayınlara web üzerinden erişerek ilgili yöntem ve pratiğin ağırlık ve baskınlığını etkilemektedir. Bu çalışma ile UYY yöntem ve pratiklerinin ortaya çıkış zamanları, tüm kütle içerisindeki oranları ve endüstriyel kullanıcının ihtiyacı ile zamansal izdüşümleri, elektronik kütüphaneler ve web arama bilgileri karşılaştırılarak incelenmiştir.

Anahtar Kelimeler: UYY, Uygulama yaşam döngüsü yönetimi, yazılım yaşam döngüsü modelleri, sistematik eşleme

Systematic Mapping Study for Software Lifecycle Models

Yagup Macit[1], Ahsen İkbal Aytekin[1], Çağdaş Üsfekes[1]

[1]HAVELSAN Askeri Yazılımlar Grup Müdürlüğü
06510 Ankara, Türkiye
{ymacit, ayergok, cusfekes}@havelsan.com.tr

Abstract. Activities that cover all phases of application software from idea to maintenance phase are defined as Application Lifecycle Management (ALM) process. The methods and practices in the process vary evolutionarily according to current needs and possibilities. The evolution of software development methods and practices continues under the collaboration and competition of

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

industry and academic circles. The life cycle of each evolutionary method and practice may vary according to its coverage and level of abstraction. Again, the weight and dominance of each method and practice within all methods and practices may vary. In addition, industrial software developers can access academic publications on the web, depending on their method and practical needs, and influence the weight and dominance of the relevant method and practice. In this study, the emergence times of the ALM methods and practices, their proportions in the whole group and the needs of the industrial user and their time-based projections are examined by comparing the electronic libraries and web search information.

Keywords: ALM, Application Lifecycle Management, Software lifecycle models, Systematic mapping

1 Giriş

Bilgisayar yazılımlarının geliştirilmesi ve yaşatılması, yazılımın tüketicisi ve üreticisi farklılaşmaya başladıktan sonra, yönetilmesi gereken zorlu bir süreç olarak kendisini götermiştir. Büyük yazılım sistemlerinin üretilmesinde ortaya çıkan karmaşıklığa, Şelale yöntemi [1] ile verilen karşılık, evrimsel bir yolculuk ile süregelmektedir.

Bilgisayar yazılımlarının yaşamı, her hangi bir ürünün yaşamına benzer şekilde, sıralı aşamalar içermektedir. Fikir aşamasından başlayan yolculuk, tasarım, geliştirme, doğrulama, dağıtım ve bakım gibi diğer aşamalar tarafından takip edilmektedir. Fikir aşamasından başlayıp, bakımın sonlandırılması ve ürünün geri çekilmesi aşamalarının tamamı Uygulama Yaşam Döngüsü Yönetimi (UYY) olarak tanımlanmaktadır [2].

Yazılımların yaşam döngüsü üzerinde kullanıcılarının ve çalıştığı ortamların etkisi kaçınılmazdır. Başlangıçta sınırlı sayıda kullanıcısı olan MainFrame sistemler için üretilen yazılımlar, günümüzde sınırsız kullanıcısı olan mobil cihazlar için üretilebilmeketedir. Kullanıcı beklentileri ve ortam yeteneklerindeki değişim, rekabet açısından üretim ve dağıtım için sürelerin kısaltılmasını zorlamaktadır. Her döneme özgü kullancı beklentisi ve barındırma ortamının yetenek gelişimi, UYY yöntemlerinin değişimini tetikleyen temel evrim mekanizması olmuştur.

Bu çalışmada, endüstriyel kullanım ve akademik çalışma sıklığıyla önce çıkan UYY yöntem ve pratikleri genel olarak listelenerek inceleme alanı tanımlanmıştır. Akademik yayınlar incelendiğinde yöntem ve pratik listesi oluşturmak için tek bir yayın referansının yeterli olmadığı görülmüş ve çoklu referans ile liste oluşturulmuştur. Yöntem ve pratiklerinin anlaşılırlığını desteklemek ve yöntem isimlerinin tüm süreç içerisinde nereye odaklandığının bilgisini vermek için UYY kavramının aktarımı yapılmıştır. Oluşturulan UYY listesi için yöntemin ortaya çıkış yılı, yöntemin diğer yöntemler arasındaki ağırlığı ve yöntemin web aramalarına yanıt verme eğilimine odaklanan bir Sistematik Eşleme Çalışması yapılmıştır. Sistematik Eşleme Çalışması için verilerin elde edilmesi amacıyla, UYY listesinin her bir öğesi, yaygın olarak kullanılan elektronik kütüphaneler üzerinde sorgulanmıştır.

Bildirinin ikinci bölümünde, incelenen yöntemlerin kavram alanı olan Uygulama Yaşam Döngüsü yönetimi ve inceleme yöntemi olan Sistematik Eşleme Çalışması

hakkında bilgi verilmiştir. Üçüncü bölümde araştırma yönetimi ve ayrıntıları alt başlıklar halinde açıklanmıştır. Son bölümde ise yapılan eşleme çalışması genel olarak özetlenmiş ve sonraki dönemde yapılacak çalışmalar için bilgi verilmiştir.

2 Temel Bilgiler

2.1 Uygulama Yaşam Döngüsü Yönetimi

Uygulama Yaşam Döngüsü Yönetimi (UYY) bir yazılım geliştirme projesinin ortaya çıkışındaki fikir aşaması ile projenin geliştirme, dağıtım ve bakım süreçlerini kapsayan bir yaklaşımdır [2]. Bu sayede UYY, yazılım projelerinde tüm süreçlerde güçlü bir yönetim ve izlenebilirlik yeteneği sağlar [3].

UYY kavramı, UYY 1.0 yaklaşımı ile ortaya çıkmıştır. UYY 1.0 yaklaşımında bir yazılım projesindeki her disiplin için farklı araçların kullanılması ve bu araçların birbiri ile olan entegrasyonu önerilmiştir. Test, Tasarım, Geliştirme, İnşa gibi süreçleri yöneten farklı araçlar karşılıklı bilgi alışverişi yapabilir [4]. Bu entegrasyonun zor ve maliyetli olması sebebiyle UYY 2.0 yaklaşımı doğmuştur [4] [5]. UYY 2.0 yaklaşımında yazılım geliştirme süreçlerinin farklı araçlardan ve süreçlerden takip edilmesi yerine, uygulama yazılımlarının süreçlerin bütünleşik olduğu araçlar üzerinden yönetilmesi benimsenmiştir [6] [7] UYY 2.0 yaklaşımına uygun araçlar rol ve yetki tabanlı olarak çalışmaktadır. Bu yaklaşımda süreçler arası iletişim ve bilgi alışverişi sistematik olarak yapıldığından UYY 1.0 yaklaşımındaki entegrasyon maliyetleri ortadan kalkmıştır.

2.2 Yazılım Yaşam Döngüsü Modelleri

Yazılım geliştirme süreçlerinde yaşanan evrim boyunca pek çok yaşam döngüsü modeli geliştirilmiştir. Bhuvaneswari [8], Sujit Kumar Dora [9], Crnkovic [10] ve Coopera [11] çalışmalarında bu yaşam döngüsü modellerine detaylı açıklamaları ile birlikte değinmiştir. Bu çalışmada ise 22 adet yazılım yaşam döngüsü modeli üzerinde durulmuştur. Bu modeller Tablo 2.'de verilmiştir.

2.3 Sistematik Eşleme Çalışması

Kanıta Dayalı (Evidence-based) [12] kavramı öncelikle tıp sektöründe kullanılmıştır. Bu sektördeki başarılı sonuçlarının ardından diğer alanlarda da kullanılmaya başlanmıştır. Bu kavramın yazılım mühendisliğindeki ilk örneği Kanıta Dayalı Yazılım Mühendisliği (KDYM) adıyla 2004 yılında ortaya çıkmıştır. Bu çalışmada bir alanda yeni bir yöntemin geliştirildiği ve kıyaslamaların yapıldığı en eski referanslı çalışmalardan deneysel kanıt amaçlı veri toplanır. Bu veri üzerinde Sistematik Literatür Taraması (Systematic Literature Review-SLR) veya Sistematik Eşleme Çalışması (Systematic Mapping Study) ile araştırma yapılır. Sistematik eşleme çalışması genel tanım olarak, bir konuda ilk yapılan bir diğer deyişle en eski tarihli çalışmaları sınıflandırmaktır. Bu çalışmalar ilgili konu hakkında detaya inmeden üst düzey bilgiler

verir. Eşleme çalışmalarında kullanılan soruların içeriği genel amaca yöneliktir. Ayrıca kullanılan arama yöntemleri çok katı ve sınırlı olmamakla birlikte temel amaç, arama sonucunda bulunan çalışmaları sınıflandırmaktır.

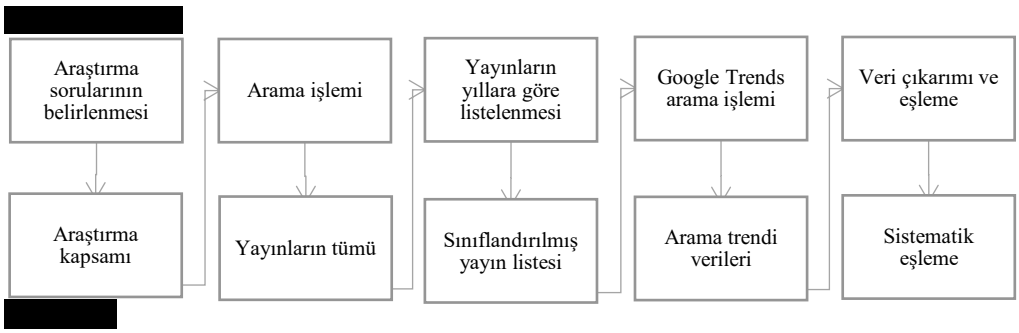
Bu çalışmada belirlenmiş olan yazılım yaşam döngüsü modeli ile ilgili yayınlanmış, konferans bildirileri, tezler, kitaplar ve makaleler incelenmiştir.

3 Araştırma Yöntemi

Literatürde kullanılan sistematik eşleme çalışması yöntemleri [13] derlenerek, bu bildiride kullanılmak üzere aşağıdaki adımlar belirlenmiştir:

- Araştırma sorularının tanımlanması
- UYY listesi ile ilgili yayınlara ulaşmak için elektronik veritabanlarında aramaların yapılması
- Elde edilen arama sonuçlarını sınıflandırmak amacıyla, her kütühanedeki her model için yayın tarihlerinin listelenmesi
- Yıllara göre her model için yapılan çalışma sayısının belirlenmesi
- Google Trends üzerinde UYY modelleri için aramaların yapılması
- Her iki arama sonuçları üzerinde veri çıkarımının yapılması ve çalışma sonuçlarının yazılması

Şekil 1. ' de Sistematik eşleme çalışması için akış diyagramı verilmiştir.



Şekil 1. Sistematik Eşleme Çalışması Adımları

3.1 Araştırma Soruları

Sistematik Eşleme Çalışmasının amacı bir konuya ilişkin yayınlanmış çalışmaları sınıflandırmaktadır. Sınıflandırma işlemi için birincil çalışmalara önceden belirlenmiş olan araştırma soruları sorularak cevaplarına göre sonuçlar elde edilmektedir. UYY modelleri eşleme çalışması için aşağıdaki araştırma soruları (AS) ilgili motivasyonlar doğrultusunda belirlenmiştir.

- AS1: Model ilk kez ne zaman görülmüştür?
  Motivasyon: Modeller arasında genetik/tarihsel devamlılığın araştırılması.

- AS2: Modelin diğer modeller içerisindeki ağırlığı / konumu nedir?
  Motivasyon: Modellerin akademik popüleritesinin gözlemlenmesi.
- AS3: Modelin endüstri kullanıcılarının web arama eğilimi ile akademik yayın eğiliminin örtüşme oranı nedir?
  Motivasyon: Modeller için akademik ve endüstriyel popularitesinin eşlenikliğinin ölçümlenmesi.

Çalışma sonucunda bu sorulara verilen cevaplar kullanılarak elde edilen sonuçlar, her soruya yanıt olabilecek şekilde raporlanmış ve değerlendirilmiştir.

3.2 Arama Yöntemi

UYY modelleri konusundaki Sistematik Eşleme Çalışmasında kullanılacak yayınlar için IEEE Xplore, ACM Digital Library, Wiley Inter Science Journal Finder, Science-Direct, Springer Link ve ISI Web of Knowledge elektronik veri tabanlarında arama yapılmıştır. Arama yöntemi olarak bu veri tabanlarının gelişmiş arama bölümleri kullanılmış ve hazırlanan sorgu cümleleri çalıştırılmıştır. Arama sonucunda UYY modelleri konusunda yayınlanan akademik çalışmalar elde edilmiştir. Araştırma sorularına göre, UYY modellerinin web arama eğilimine ulaşmak için Google Trends üzerinde modellerin yıllara göre arama trendleri CSV formatında çıkarılmış ve arama sonuçlarına eklenmiştir.

3.3 Sorgu Cümleleri

Arama yönteminde belirtilen veri tabanlarının gelişmiş arama bölümleri için mantık operatörleri kullanılarak her model için sorgu cümleleri hazırlanmıştır. Aşağıdaki tabloda UYY modellerinden örnek seçilen “V-Model” araması için sorgu cümleleri paylaşılmıştır. Diğer modeller için sorgu cümleleri “V-Model"/"V-Shaped" metinleri ile değiştirilerek benzer şekilde hazırlanmıştır. Modellerin literatürde yaygın kullanımda olan kısaltmaları, sorgu cümlelerinde “OR” operatörü kullanılarak harfe duyarlı şeklide aranmıştır. Örneğin, Extreme Programming için hazırlanan sorgu cümlesinde “XP” öbeği OR seçeneği ile kullanılmıştır.

Tablo 1. Elektronik veritabanları “V-Model” araması için örnek sorgu cümleleri

ACM	(acmdlTitle:("V-Model" "V-Shaped") OR recordAbstract:("V-Model" "V-Shaped") OR keywords.author.keyword:("V-Model" "V-Shaped")) AND recordAbstract:(Software)
IEEE	(("Publication Title":"V-Model" OR "Abstract":"V-Model" OR "Document Title":"V-Model" OR "Publication Title":"V-Shaped" OR "Abstract":"V-Shaped" OR "Document Title":"V-Shaped") AND ("Publication Title":software OR "Abstract":software OR "Document Title":software))

SienceDirect	Title, abstract, keywords: ("V-Model" OR "V-Shaped") AND "Software"
Springer	https://link.springer.com/search?dc.title="V-Model"+AND+"software"+OR+dc.abstract="V-Model"+AND+"software"+OR+dc.title="V-Shaped"+AND+"software"+OR+dc.abstract="V-Shaped"+AND+"software"&sortOrder=oldestFirst
WebOf-Konwledge	TS=("V-Model" OR "V-Shaped") AND TS=(Software)
Wiley	""V-Model" OR "V-Shaped"" anywhere and "Software" in Abstract

Wiley veri tabanı için kullanılan sorgu cümlesiyle yapılan arama sonucundaki sayısal değerler, RSS çıktısındaki değerler ile örtüşmediğinden bu veri tabanından çıkan sonuçlar değerlendirmeye alınmamıştır.

Google eğilimleri için https://trends.google.com.tr/trends adresindeki metin kutusuna model isimleri tırnak içerisinde yazılarak arama yapılmıştır. Aramalar, eğilim verilerinin erişilebilir olduğu 2004 senesinden itibaren dünya çapındaki sonuçları içerecek şekilde gerçekleştirilmiştir.

3.4 Veri Çıkarımı

Elektronik veri tabanları üzerinde gerçekleştirilen arama sonuçlarına ait birincil yayınlar, her model için yıllara göre yapılan yayın sayısına göre sınıflandırılmıştır.

Google Trends üzerinde yapılan arama sonucuna göre 2004 yılından itibaren aylık bazda arama trendlerinin 100 üzerinden ölçeklenmiş formatına erişilmiştir. Trend aramasında gerçek veriler yerine ölçeklenmiş verilere erişilebildiğinden, "AS3" için yayınların yıllara göre olan sayıları da 100 üzerinden ölçeklenerek kullanılmıştır.

Sonuç kümesindeki yayınlara yöneltilen araştırma sorularına göre aşağıda belirtilen hedef çıktılar elde edilerek sonuçlar bölümü için kullanılacak veriler hazırlanmıştır.

Araştırma Soruları	**Hedef çıktı**
AS1. Model ilk kez ne zaman görülmüştür?	UYY modelinin doğum yılı
AS2. Modelin diğer modeller içerisindeki ağırlığı / konumu nedir?	UYY modelinin yayın yoğunluğu
AS3. Modelin endüstri kullanıcıların web arama eğilimi ile akademik yayın eğiliminin örtüşme oranı nedir?	UYY modelleri için yayın üretimi ile web aramaları arasında, ilişkinin varlığı

4 Araştırma ve Bulgular

4.1 Araştırma Verisi

UYY Yöntemlerinin araştırılması için kullanılan IEEE Xplore, ACM Digital Library, ScienceDirect, Springer Link ve ISI Web of Knowledge elektronik veri tabanları üzerinden elde edilen veriler, Tablo 2.'de yöntem adı ve elektronik veri tabanı boyutlarına göre bulgu sayısı şeklinde verilmiştir.

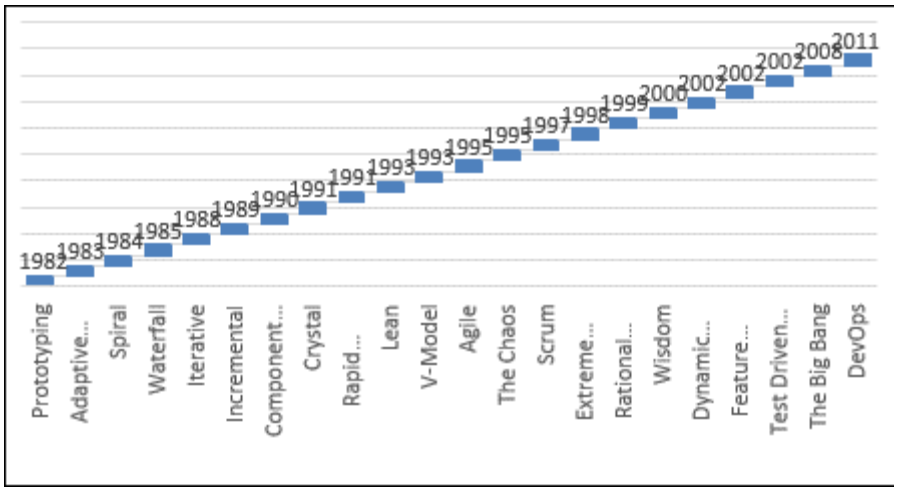
Tablo 2. Yöntem Türlerine Göre Arama Sonuçları

Yöntem/Kütüphane	ACM	IEEE	Science Direct	Springer	WoS	Toplam
Adaptive Software Development	5	20	82	1	29	137
Agile	1194	2885	570	1800	4839	11288
Component Based Software Development	98	548	107	37	971	1761
Crystal	48	2	7	13	9	79
DevOps	133	163	22	100	202	620
Dynamic System Development	3	2	3	5	6	19
Extreme programming	298	419	193	151	529	1590
Feature Driven Development	3	12	18	16	30	79
Incremental	19	26	35	24	278	382
Iterative	10	13	61	13	256	353
Lean	142	303	248	224	1094	2011
Prototyping	17	3226	47	613	4910	8813
Rapid Application Development	21	75	24	13	148	281
Rational Unified Process	29	99	16	12	165	321
Scrum	182	449	87	214	902	1834
Spiral	54	378	267	9	1673	2381
Test Driven Development	128	190	29	89	300	736
The Big Bang	3	5	5	5	7	25
The Chaos	9	34	102	18	62	225
V-Model	28	87	46	15	250	426
Waterfall	87	256	62	20	410	835
Wisdom		148	36	59	287	530

Toplam	**2511**	**9340**	**2067**	**3451**	**17357**	**34726**

4.2 İlk Yayın Yılı İncelemesi

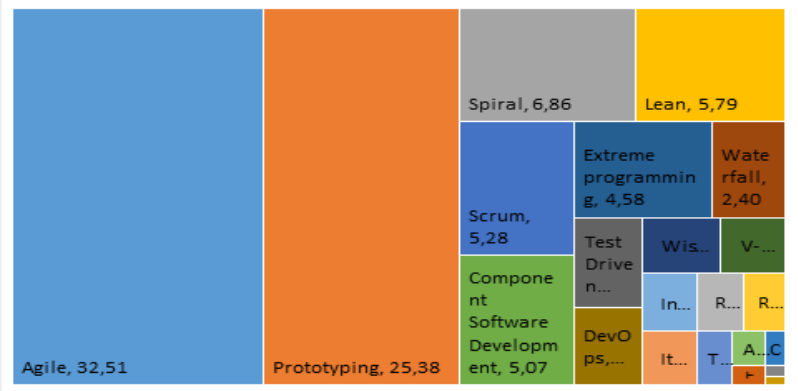
Elektronik veri tabanları üzerinden elde edilen ve Tablo 2.'de verilen yayınlar, türlerine göre birleştirilerek yayın yılı üzerinden inceleme yapılmıştır. Her bir yayın türü için en küçük yayın yılı bulunmuştur. Elde edilen bilgi, yayın türü ve en küçük yayın yılı grafiği olarak Şekil 2.'de verilmiştir.



Şekil 2. Yöntemlerin İlk Bulgu Yılları

4.3 Yöntem Türü Yoğunluğu İncelemesi

Elektronik veri tabanları üzerinden elde edilen ve Tablo 2.'de verilen yayınlar içerisinde yöntem türü toplamları, genel toplama bölünerek her bir yöntem türü için oransal ağırlık değeri hesap edilmiştir.



Şekil 3. Yöntem Türlerini Ana Kütle İçerisindeki % Yerleşim Yoğunlukları

Elde edilen yayın türü ve ana kütle içerisindeki oran bilgisi grafik olarak Şekil 3. ’de verilmiştir.

4.4 Eğilim Krşılaştırması

Endüstriyel bilişim profesyonellerinin başvuru kaynaklarından bir olan Google arama motoru, geçmiş döneme ait arama istatistiklerini, eğilim bilgisi şeklinde paylaşmaktadır. Arama istatistikleri, 2004 yılından sonrası için ve sorgu sayıları yerine 100 üzerinden eğilim bilgisi olarak alınabilmektedir.

İlk Yayın Yılı İncelemesi başlığında, yayınların yöntem türlerine göre birleştirilmesinden sonra, her bir yöntem için 2004 ve sonrası yıllara göre yayınlar sayılmıştır. Her bir yöntem için elde edilen yıllara göre yayın sayıları, en yüksek yıl değeri 100 kabul edilerek diğer yıllar bu yıla oranlanmış ve eğilim bilgisi hesaplanmıştır.

Google ve elektronik veri tabanlarının zamana dayalı eğilim serilerinin karşılaştırılması için t-testi [14] incelenmiştir. Karşılaştırmaya konu olan Google ve elektronik veri tabanlarının zaman serileri birbirinden bağımsız ve faklı varyanslara sahip olacağı için t-testinin, bağımsız ve farklı varyansları örneklemlere genelleştirilmiş hali [15] kullanılmıştır.

Eğilim benzerliği t-testi için sıfır hipotez (H_o), her bir yöntem türü için Google eğilim ortalaması ile elektronik veri tabanı eğilim ortalaması arasında “%95 olasılıkla anlamlı bir fark yoktur” yargısı ile kurulmuştur. Karşıt hipoteze (H_1) göre ortalamalar arasında fark vardır. Hipotezler, matematik modeli olarak aşağıdaki gösterimle tanımlanmıştır.

H_o: $\mu_{google} = \mu_{veritabanı}$

H_1: $\mu_{google} \neq \mu_{veritabanı}$

Hipotezlerin sınanması işlemi, Excel Analysis ToolPak ile “*t-Test: Two-Sample Assuming Unequal Variances*” seçeneği kullanılarak yapılmıştır. Excel ile her bir yöntem için Google ve veri tabanı temelinde ortalama, varyans ve t-test değerleri hesaplanmıştır. Hesaplanan t-test değerlerin 0,05 güven aralığı değerinden büyük olması durumunda hipotez kabul edilmiştir. Hesaplama sonuçları Tablo 3.’de verilmiştir.

Tablo 3. Google ve Veri Tabanı Eğilim Serileri için İstatislikler

	Ortalama		Varyans			
Yöntem	**Google**	**Veri Tabanı**	**Google**	**Veri Tabanı**	**t-Test**	**Sonuç**
Adaptive Software Development	38,39	50,00	587,809	634,074	0,194157	KABUL
Agile	62,03	55,40	414,780	653,293	0,424139	KABUL
Component Based Software Development	21,99	49,02	717,883	685,197	0,007159	RET
Crystal	44,84	50,78	484,255	759,766	0,505705	KABUL
DevOps	23,64	21,77	1138,081	1149,876	0,876749	KABUL
Dynamic System Development	25,77	33,33	551,054	1185,185	0,473963	KABUL
Extreme programming	21,52	56,68	734,717	539,017	0,000464	RET

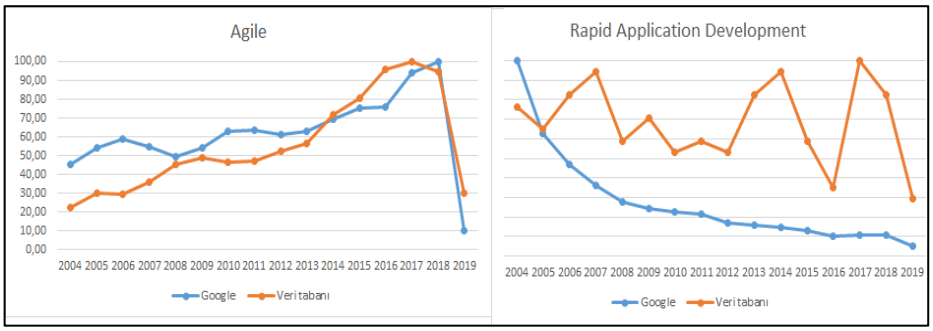
Feature Driven Development	33,00	62,50	666,300	894,558	0,005642	RET
Incremental	59,63	67,31	192,382	489,152	0,250521	KABUL
Iterative	42,07	54,46	652,294	562,143	0,165173	KABUL
Lean	62,12	45,21	286,130	802,374	0,051183	KABUL
Prototyping	43,65	74,69	354,329	430,399	0,000117	RET
Rapid Application Development	27,48	68,38	599,832	423,875	0,000018	RET
Rational Unified Process	33,29	43,75	725,159	624,456	0,263897	KABUL
Scrum	56,62	43,02	529,510	1046,831	0,181687	KABUL
Spiral	49,57	67,35	311,832	349,884	0,009635	RET
Test Driven Development	70,54	71,17	219,068	435,961	0,922396	KABUL
The Big Bang	59,74	22,32	387,721	951,531	0,000381	RET
The Chaos	29,44	41,44	595,904	715,535	0,195372	KABUL
V-Model	65,60	54,44	363,259	839,506	0,209568	KABUL
Waterfall	55,41	48,38	304,819	639,977	0,368578	KABUL

Yapılan hesaplama işlemleri için karşılaştırma örneği olabilecek “Agile” ve “Rapid Application Development” yöntemleri için detaylı t-test değerleri Tablo 5.’de verilmiştir.

Tablo 4. Karşılaştırmalı t-test Örneklemesi

t-Test: İki Örnekli Eşit Olmayan Varyans Varsayımı	Agile		Rapid Application Development	
	Google	***Veri Tabanı***	***Google***	***Veri Tabanı***
Ortalama	62,029159	55,404020	27,480850	68,382353
Bilinen Varyans	414,779592	653,292942	599,831676	423,875433
Gözlem	16	16	16	16
Öngörülen Ortalama Fark	0		0	
df	29		29	
t Stat	0,81087638		-5,113419259	
P(T<=t) tek uçlu	0,21202089		9,25338E-06	
t Critical tek uçlu	1,69912703		1,69912703	
P(T<=t) iki uçlu	**0,42404178**		**1,85068E-05**	
t Critical iki uçlu	2,04522964		2,04522964	

Karşılaştırması yapılan “Agile” ve “Rapid Application Development” yöntemlerinin eğilim grafikleri Şekil 4. ’de verilmiştir.



Şekil 4. Karşılaştırmalı Eğilim Grafikleri

5 Sonuçlar

Uygulama yazılımlarının yaşam döngüsü yönetimi için endüstri ve akademi çevrelerinde yöntem arayış ve üretme çalışmalarının varlığı gözlemlenmiştir.

Akademik çalışma yoğunluğu bulgularına bakıldığında, Agile 11301 değeri ile en çok, Dynamic System Development 19 değeri ile en az, çalışılan yöntem olduğu gözlemlenmiştir.

Yöntemlerin ilk yayın yılı bulgularına bakıldığında, yaşam döngüsü yönetimi ihtiyaçlarına yanıt vermek için şelale ve yinelemeli türlerinin kendi aralarında sürgit devam eden evrimsel bir süreklilik gözlemlenmiştir.

Yöntemlerin ana kütle içerisindeki oransal ağırlıklarına bakıldığında, homojen bir dağılım olmadığı ilk 5 yöntemin %75'lik bir oranı oluşturduğu görülmüştür. Bu durum, ihtiyaca yanıt olarak üretilen yöntemin kullanım karşılığı bulamaması ve sönümlenmesi olarak açıklanabilir.

Yöntemler için endüstri ve akademik eğilimin, sayısal olarak % 67 oranında, oransal ağırlık olarak %57 oranında örtüştüğü gözlemlenmiştir. Bu durum, oransal ağırlığı az olan yeni yöntemler için örtüşmenin daha yaygın olması şeklinde açıklanabilir. Ek olarak, eğilimlerin eşleşmediği ve sıfır hipotezinin reddedildiği yöntemler için endüstriyel bilinirliğin belirleyici olduğu gözlemlenmiştir.

Önümüzdeki dönemde, UYY yöntemleri arasında %90 oransal temsil gücüne sahip olan yöntemler için zamana dayalı mimari stil ile eşleme araştırması planlanmaktadır.

Teşekkür. Yazarlar HAVELSAN yönetimine çalışmaya verdiği destek için teşekkür ederler.

6 Kaynakça

1. W.W.Royce, "Managing the Development of Large Software Systems," *Proceedings of IEEE WESCON 26 (August): 1–9,* pp. pp.328-338, 1970.
2. D. Chappell, What is Application Lifecycle Management?, Chappell & Associates, 2008.

3. E. T. Y. M. B. T. A.İ. Aytekin, "Uygulama Yaşam Döngüsü Yönetimi - Sistematik Eşleme Çalışması,"*UYMS*, İzmir, 2015.
4. J. V. A. Kääriäinen, "Impact of Application Lifecycle Management – A Case Study," in *International Conference on Interoperability of Enterprise (IESA)*, Berlin, 2008.
5. C. e. a. Schwaber, "The Changing Face of Application Lifecycle Management," in *Forrest-er Research*, 2006.
6. K. Shaw, "Application Lifecycle Management for the Enterprise," in *Serena Software*, 2007.
7. J. K. a. A. Välimäki, "Applying ALM for the Development of Complex Systems : Experiences from the Automation Industry," in *EuroSPI*, 2009.
8. S. P. T Bhuvaneswari, "A Survey on Software Development Life Cycle Models," in *IJCSMC*, 2013.
9. P. D. Sujit Kumar Dora, "Software Development Life Cycle (SDLC) Analytical Comparison and Survey on Traditional and Agile Methodology," *Abhinav National Monthly Refereed Journal of Research in Science & Technology*, 2013.
10. I. Crnkovic, "Component-based software engineering — new challenges in software development," *Proceedings of the 25th International Conference on Information Technology Interfaces*, 2003.
11. S. R. L. S. K.Coopera, "Integrating visual goal models into the Rational Unified Process", *Journal of Visual Languages & Computing, Volume 17, Issue 6, Pages 551-583,* 2006.
12. B. A. D. T. a. J. M. Kitchenham, "Evidence-Based Software," in *Proc. Of the 26th International Conference on Software Engineering (ICSE)*, Scotland, 2004.
13. K, Petersen; R, Feldt; S, Mujtaba; M, Mattsson, "Systematic mapping studies in software engineering," in *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, 2008.
14. Student, "The Probable Error of a Mean," *Biometrika,* vol. 6, no. 1, pp. 1-25 https://doi.org/10.1093/biomet/6.1.1, Mart 1908.
15. B. L. Welch, "The Generalızatıon Of 'Student's' Problem When Several Dıfferent Populatıon Varlances Are Involved," *Biometrika,* vol. 34, no. 1-2, pp. 28–35, https://doi.org/10.1093/biomet/34.1-2.28, Ocak 1947.
16. P. Feiler and W. Humphrey, "Software Process Development and Enactment:Concepts and Definitions," Software Engineering Institute, 1992.

Model Tabanlı Geliştirme Teknolojisi Kullanılarak Geliştirilen Hava Aracı Yazılımlarının Sertifikasyon Sürecine Yönelik Öneriler

Tuğba Saraç

Türk Havacılık Uzay Sanayii, Havacılık Bulvarı No:17, Ankara, Türkiye
tsarac@tai.com.tr

Özet. Teknolojik gelişmelerin hızla ilerlemesi, ihtiyaç duyulan ürünler ile ilgili alternatiflerin çoğalmasını ve müşterilerin çok sayıda alternatife daha kolay bir şekilde erişmesini sağlamıştır. Bu durum üreticiler arasında sıkı bir rekabetin oluşmasına neden olmuştur. Artan rekabet koşullarında üreticiler, bir adım öne geçebilmek için müşteri isteklerine daha az zamanda daha hızlı cevap verebilecek yeni teknoloji arayışlarına gitmişlerdir. Bu teknolojilerden birisi de, üreticilere sağladığı takvim ve maliyet avantajı nedeni ile havacılık sanayiinde son dönemlerde öne çıkan model tabanlı geliştirme teknolojisidir. Model tabanlı geliştirme teknolojisi, geleneksel yazılım geliştirme yaklaşımına alternatif olarak sunulan sistem ve yazılım seviyesindeki faaliyetlerin iç içe geçtiği yeni bir geliştirme teknolojisidir. Bu teknolojinin telekomünikasyon gibi emniyet kritik olmayan yazılımların geliştirilmesinde kullanımı daha eski olsa da, emniyet kritik olan ve sertifikasyon içeren hava aracı yazılımlarının geliştirilmesinde kullanımı oldukça yenidir. Bu çalışmada, MTG teknolojisinin havacılık sektöründe kullanımı, sağladığı önemli avantajlar ve ürün geliştirime süreçlerinde dikkat edilmesi gereken hususlar sunulmuştur. İlave olarak model kapsama analizi ile yapısal kapsama analizi arasındaki fark bir örnek üzerinden anlatılmıştır.

Anahtar kelimeler: Model Tabanlı Geliştirme, Hava Aracı Sertifikasyonu, Model Kapsama Analizi.

Suggestions for the Certification Process of Airborne Software Developed Using Model-Based Development Technology

Tuğba Saraç

Türk Havacılık Uzay Sanayii, Havacılık Bulvarı No:17, Ankara, Türkiye
tsarac@tai.com.tr

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Abstract. The rapid improvements of technological developments has led to increase alternatives and to enable customers to access these alternatives more easily and quickly. This has led to an increase in competition among companies. Increasing competition conditions have led companies to seek alternative technologies to produce products in less time and less cost. One of these technologies is the model-based development technology, which is very popular due to the time and cost advantages it provides to companies. Model-based development technology is a new development technology used as an alternative approach to the traditional software development approach. Although the use of this technology in the development of non-safety critical software such as telecommunications is older, the use of this technology in the development of safety critical software such as aircraft software is relatively new. In this study, the use of this technology in the aviation, the advantages it provides, the issues that need to be considered during the development processes are explained.

Keywords: Model-Based Development, Airborne Software Certification, Model Coverage Analysis.

1 Giriş

Son yıllarda hızla gelişen teknolojiye paralel olarak, havacılık alanında kazanılan bilgi birikimi, ürün geliştirme süreçlerinde kullanılan yetenekli araçlar, rakip firmaların ürettiği ürün çeşitliliği ve daha bilinçli müşteri profili sayesinde, daha kabiliyetli ürünler daha ekonomik bir şekilde tasarlanarak üretilebilmektedir. Önceki dönemlerde donanımlar aracılığı ile icra edilen birçok hava aracı fonksiyonu günümüz hava araçlarında yazılım ile çok daha hızlı, etkin ve ekonomik bir şekilde icra edilebilmektedir.

Bununla birlikte, daha kabiliyetli hava araçları beraberinde daha karmaşık sistemleri de getirmiştir. Projelerde sıklıkla görülen takvim ve maliyet baskısı ile birlikte, bu karmaşık sistemlerde çalışan yazılımların geliştirme ve doğrulama süreçleri de hataya daha açık hale gelmiştir. İlave olarak, geleneksel yazılım geliştirme yaklaşımı ile geliştirilen emniyet kritik yazılımları doğrulamak için harcanan süre toplam proje süresinin %50'si ile %70'i arasındadır [1]. MTG teknolojisi ile sistemin hataları erken aşamalarda yakalanabildiği ve kod otomatik olarak üretilebildiği için ürünler daha kısa zamanda geliştirilebilmekte ve modelleme araçlarının simülasyon avantajları da kullanılarak sistem davranışı, farklı koşullarda henüz kod dahi üretilmeden projenin erken aşamalarında test edilebilmektedir [1].

MTG teknolojisinin sertifikasyon içeren hava araçlarında kullanımı ile ilgili kurallar ilk kez 2008 yılında Avrupa Sivil Havacılık Otoritesi (EASA) tarafından yayımlanmıştır [2]. 2011 yılında RTCA (Radio Technical Commission for Aeronautics) tarafından yayınlanan DO-178C dokümanının DO-331 eki ile bu kurallar son halini almıştır. DO-178C dokümanı, hava aracında kullanılacak yazılımların, yaşam döngüsü süreçlerini, bu süreçlerin amaçlarını ve üretilecek verileri anlatmaktadır [3].

MTG teknolojisi telekomünikasyon veya sertifikasyon içermeyen hava araçlarında kullanımı daha eski olsa da, sertifikasyon içeren hava aracı yazılımlarında kullanımı yenidir. Havacılık sektöründe ortalama proje sürelerinin 3-4 yıl olduğu düşünülürse,

kuralları ilk kez 2008 yılında tanımlanan MTG teknolojisi ile geliştirilerek sertifiye edilen çok fazla hava aracı olmaması normaldir. MTG teknolojisi, Airbus A380 uçağında, uçuş kontrol sistemleri, oto pilot, uçuş ikaz sistemleri, yakıt yönetimi, iniş takımı, buzdan korunma sistemi, elektriksel yük analizi sistemlerinin ve bu sistemlerin içinde yer alan yazılımlarında geliştirilmesinde kullanılmıştır [4].

2 MTG Teknolojisi ve Başlıca Avantajları

Model, analiz, doğrulama, simülasyon, kod oluşturma veya bunların kombinasyonu için kullanılmak üzere, sistemden yazılıma akacak öğelerin soyut gösterimidir [5]. MTG teknolojisi ise, sistem davranışlarının ve/veya yazılım gereksinimlerinin, modellerle ifade edilebildiği bir geliştirme teknolojisidir [5]. MTG teknolojisi ile modelleme aracında geliştirilen sistemin doğrulama çalışmalarına, daha yazılım kodu üretilmeden başlanabilmekte, sistemdeki hatalar ve olası performans iyileştirme ihtiyaçları projenin erken fazlarında tespit edilebilmektedir. MTG teknolojisinin bir başka avantajı da, otomatik kod üretmeye imkan vermesi ve üretilen kodun hava araçlarında kullanılarak sertifiye edilebilmesidir. MTG teknolojisi ile otomatik üretilen kod, DO-331 rehber dokümanına uygun geliştirilirse hava araçlarında kullanılarak sertifiye edilebilir. Bu teknolojide, sistem seviyesinden hazırlanan modellerden direkt kaynak kod üretilebildiği için, üst seviye ve alt seviye yazılım gereksinimleri, yazılım mimarisi ve kodun oluşturulmasına gerek kalmaz. Bu verileri oluşturmak ve gözden geçirmek için ayrılan süre, sistemin farklı koşullardaki davranışını ve uçuş emniyetini etkileyebilecek durumları incelemek için kullanılabilir. Ayrıca, bu aşamalarda oluşabilecek insan hataları da azaltılmış olur [4].

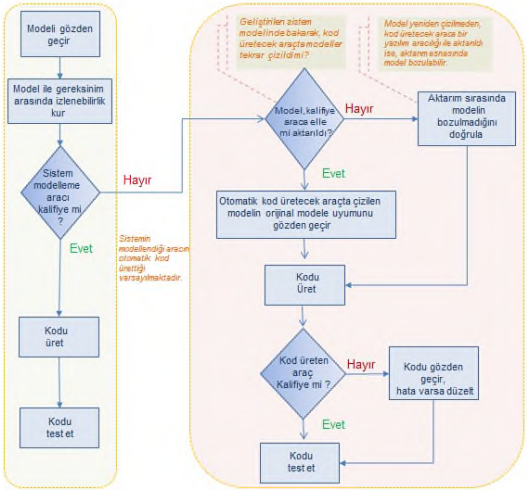
2.1 MTG Teknolojisinde Dikkat Edilmesi Gereken Hususlar ve Öneriler

MTG teknolojisinde kod genellikle modelden otomatik olarak üretildiği için yazılımın davranışı modele bağlıdır. Model sistemi tam olarak temsil etmiyor veya hata içeriyor ise, modelden üretilen yazılım da beklenen fonksiyonu icra etmez. Bu teknolojide geliştirilecek modelin anlatan yazılı gereksinimler mutlaka olmalıdır [2]. Model, bu gereksinimlere göre geliştirilmeli ve soru listeleri kullanılarak tecrübeli sistem/yazılım mühendisleri tarafından yeterli sürede gözden geçirilmelidir.

MTG teknolojisi ile üretilecek yazılım hava araçlarında kullanılacak ise bu araçların kalifiye olması gerekir [3]. İnsan tarafından yapılan bir yazılım aktivitesi, araç tarafından yapıldığı için insan tarafından tekrar yapılmıyor ise veya araç aktiviteyi otomatikleştiriyor ise ve aracın ürettiği çıktı gözden geçirilmiyor ise bu aracın kalifiye edilmesi gereklidir. Örneğin bir yazılım testinin geçti / kaldı kararını, beklenen değer ile hesaplanan değeri karşılaştırarak bir araç veriyor ise, bu aracın kalifiye edilmelidir.

Sistem mühendislerinin modelleme yaptığı araçlar ile yazılım mühendislerinin otomatik kod üretimi için kullandıkları araçlar genellikle farklıdır. Sistem mühendislerinin kullandığı araçların kalifiye edilmesi zor ve pahalıdır. Geliştirilecek sistemin ve yazılımın yapısına göre modelleme, kalifiye olmayan ve dolayısıyla ürettiği kod hava ara-

cında kullanılamayan araçlarda da yapılabilir. Geliştirilen modelden otomatik kod üretilecek ise, modellerin kalifiye bir araca aktarılması ve aktarımın sorunsuz olduğunun doğrulanması gerekir. Şekil 1'de sertifikasyon içeren projelerde MTG teknolojisinin kalifiye olan/olmayan araçlar ile kullanımlarına yönelik tavsiyeler verilmiştir.



Şekil 1. Araçların Kalifikasyon Durumuna Göre Yapılabilecek Aktiviteler

MTG teknolojinin kullanımında dikkat edilmesi gereken bir diğer önemli nokta da modelin konfigürasyon ve değişiklik yönetim süreçlerdir. Modellerin nasıl isimlendirileceği, modelin geliştirildikleri gereksinimler ile modelden üretilen kod arasındaki uyumluluğun nasıl takip edileceği gibi konular yazılım planlarında tanımlanmalıdır.

Yazılımın hava aracında yükleneceği hedef ortam ile simülasyon ortamı (kullanılan donanım, işlemci, işletim sistemi gibi) tamamen aynı olmayabilir. Bu nedenle, kaynak kodun hedef bilgisayar ile uyumluluğu, modellerin model standardına uyumluluğu ve kurulan izlenebilirliklerin doğruluğu model simülasyonu ile gösterilemez [5].

Organizasyondaki araç alt yapısına bağlı olarak, modelden otomatik kod üretimi gibi, simülasyon durum ve prosedürleri de modelden otomatik olarak üretilebilir. Ancak burada dikkat edilmesi gereken önemli bir husus vardır. Hem kaynak kod, hem de simülasyon durum ve prosedürleri aynı modelden üretilirse gözden geçirme esnasına tespit edilemeyen modeldeki olası bir hata, hatalı modelden otomatik olarak üretilmiş simülasyon prosedürleri ile yakalanamayabilir. Bu nedenle, simülasyon test durum ve prosedürleri modelden otomatik olarak üretilse bile, bu prosedürlerin modelin üzerinde olan yazılı gereksinimler ile mutlaka izlenebilirlikleri kurulmalıdır.

2.2 Yapısal Kapsama Analizi ve Model Kapsama Analizi Karşılaştırması

Yapısal Kapsama Analizi (YKA) yazılım test koşularından sonra, yazılım testi ile kod içinde çalışmayan kod blokları olup olmadığını tespit edebilmek için kaynak kod üzerinde yapılan analize denir [3]. Model kapsama analizi (MKA) ise, en seviyedeki

yazılı gereksinimlerine göre hazırlanan simülasyon prosedürleri ile yapılan testlerde çalıştırılamayan model elemanı olup olmadığını tespit etmek için kullanılır [5]. Çalıştırılamayan model elemanı demek, doğrulanamayan sistem gereksinimi demektir. MKA analizinin asıl amacı, çalıştırılamadığı için doğrulanamayan sistem gereksinimlerini tespit etmektir [5]. YKA kaynak kod üzerinden yapılırken, MKA model üzerinden yapılır. Ürün geliştirme yaşam döngüsüne bakıldığında MKA, YKA'dan çok daha erken aşamada yapılmaktadır. Bu sayede eksik bir kapsama var ise, bu eksiklik geleneksel yazılım geliştirme yönteminde yazılım testlerinden sonra yapılabilen YKA aşamasında tespit edilirken MTG teknolojisinde modeller geliştirildikten sonra tespit edilebilir. Kavram olarak birbirine yakın olan bir iki analizlerin farkını açıklamak üzere aşağıda bir örnek verilmiştir. Verilen örneğin temel amacı, MKA'dan farklı olarak YKA'nın geliştirilen/üretilen koda göre değişebileceğini vurgulamaktır.

Bir hava aracındaki en önemli fonksiyonlardan birisi, pilota hava aracının yön ve durum bilgisini gösterme fonksiyondur. Kritik fonksiyonlar genellikle olası cihaz arızası, sensör arızası gibi arızalara karşı sistem seviyesinde yedeklenir. Bu örnekte, hız bilgisini sağlayan 3 farklı cihaz olduğu, bu cihazlardan gelen ve limit içinde olan hız verilerinin ortalaması ile hava aracının hızı hesaplanmıştır. Modele ait yazılı gereksinimlerden birisi aşağıda verilmiştir:

Aşağıdaki koşullar sağlandığında;
  a. Cihaz #1 geçerli bir veri sağladı ise
  b. Cihaz #2 geçerli bir veri sağladı ise
  c. Cihaz #3 geçerli bir veri sağladı ise

aşağıdaki işlemler yapılacaktır;
  a. Ortalama_Hiz değerine, geçerli verilerin ortalamasını ata
  b. Hesaplanan Ortalama_Hiz değerini ekranda göster

Bu cihazların normal koşullarda [0,100] değerleri arasında veri gönderdiği ve bu aralıkta olan verilerin "geçerli", onun dışındaki verilerin ise, "geçersiz" olduğu varsayılmıştır. Böyle bir örnekte geliştirilen doğru bir model, limitler içinde gelen girdiler ile simüle edildiğinde tüm model elemanları çalışacağı için %100 model kapsaması sağlanacaktır. MTG teknolojisinde kod genellikle otomatik üretilse bile, kodun el ile geliştirildiği MTG uygulamaları da mevcuttur (Bknz. DO-331, Tablo 1-1, Example 3). Bu örneğe ait Şekil 2.'de verilen kod yazılmış olsun. Kod incelendiğinde "else" bloğunda, sensörlerin tamamından geçerli bir değer gelmediği durumlar için hız değerinin sıfır olarak atandığı görülmektedir. Bu kod, yazılım mühendisi tarafından sensörlerden geçerli bir veri gelmediği durumda sistemin davranışını tanımlamak üzere eklenmiş bir kod olup, bu davranış gereksinimde tanımlı değildir. Bu nedenle gereksinim bazlı hazırlanan simülasyon prosedürlerinde bu senaryoya yönelik adım olmaz. Dolayısıyla limit içinde verilen test girdileri ile yapılan testler ile "else" bloğu test edilememiş, bir başka deyişle eksik yapısal kapsama oluşmuş olur. Bu örnekten de görülebileceği gibi, %100 model kapsaması sağlansa bile yapılan kodlamaya göre %100 yapısal kapsama sağlanamamış olabilir. YKA ile test edilmemiş kod blokları tespit edilerek gerekli düzeltmeler yapılmalıdır. Aksi halde, test edilmemiş kod blokları uçuş emniyetini olumsuz yönde etkileyebilir.

```
{
   double calcValue = 0.0f;
   if (validity1 || validity2 || validity3)
   {
      calcValue = (value1 * validity1 + value2 * validity2 + value3 *
validity3)/(validity1 + validity2 + validity3);
      *value = calcValue;
      return true;
   }
      else
   {
      *value = calcValue;
      return false;
   }
}
```

Şekil 2. Ortalama hız değerini hesaplayan örnek kod

3 Sonuç

Son yıllarda oldukça popüler olan MTG teknolojisi ile geliştirilen hava aracı yazılımları, DO-331 rehber dokümanında verilen kurallar sağlandığında hava araçlarında kullanılarak sertifiye edilebilmektedir. Bu çalışmada, kullanıcılarına takvim ve maliyet avantajı sağlayan bu teknolojinin kullanımı ile ilgili önemli bulunan konular kısaca açıklanmıştır. MTG teknolojisi kullanımı ile ilgili karar verirken, yazılımın sertifikasyon ihtiyacı, emniyet kritiklik seviyesi, yazılım ve sistem ekibinin kullanılacak araçlar ve DO-331/DO-178C ile ilgili tecrübesi, geliştirilecek yazılımın yapısı, karmaşıklığı, şirketin sahip olduğu araç altyapısı ve bu araçların kalifikasyon durumları değerlendirilmelidir. Özellikle daha önce bu teknoloji sertifikasyon içeren projelerde kullanılmadı ise, mümkünse danışmanlık alınması tavsiye edilmektedir. Projenin başında DO-331 dokümanında belirtilen amaçların ve yaşam döngüsünde karşılaşılabilecek olası problemlerin yazılım ve sistem ekibi ile birlikte tartışılması ve proje takvimine MTG teknolojisi ile ilgili tüm aktivitelerin (model gözden geçirme aktivitesi ve MKA aktivitesi gibi) eklenmesi tavsiye edilmektedir.

Kaynakça

1. Mats P.E. Heimdahl, Safety and Software Intensive Systems: Challenges Old and New, IEEE, 2007
2. EASA, Software Aspects of Certification, SWCEH-002 Issue 1, May 2008
3. RTCA, “DO-178C Software Considerations in Airborne Systems and Equipment Certification”, 2011.
4. Rierson L, “Developing Safety-Critical Software”, pp. 340-380, January 2013
   RTCA, “DO-331 Model-Based Development and Verification Supplement to DO-178C and DO-278A”, 2011

Gri Kurt Optimizasyonu (GWO) Algoritması İle Finans Sektörü İçin Kredi Verilebilirlik ve Sınıflandırma Çalışması

Ali TUNÇ – Kuveyt Türk Katılım Bankası, Ar-Ge Merkezi Konya
Feritpaşa Mah. Kule Caddesi Kule Plaza Kat: 9 No: 1 - 42060 Selçuklu / KONYA
ali.tunc@kuveytturk.com.tr

Özet. Bilgi teknolojilerindeki gelişmelerle birlikte, banklar, müşterilerinin kredi taleplerini etkili analitik yöntemler ve risk analizleri ile değerlendirebilmektedirler. Kredi skorlama sistemleri olarak adlandırılan yazılım ürünleri genel olarak daha önce belirlenen kredi faktörlerine göre müşterinin verilerinin toplanması, elde edilen verinin çeşitli istatistiksel veya makine öğrenmesi teknikleriyle işlenmesi ve kredi risk analizinin yapılarak nihai kredi kararının belirlenmesi aşamalarından oluşur. Bu modeller dâhilinde müşterilerin geçmiş banka hareketleri işlenerek kredi kararı verilebilmektedir. Yapılan çalışma ile müşteriye ait değişken kümelerinden oluşan veriler, makine öğrenmesi teknikleriyle işlenerek, müşteriye ait kredi değeri belirlenmeye çalışılmıştır. Gri Kurt Optimizasyonu ile sınıflandırma problemi olarak çalışma odağı oluşturulmuştur. Elde edilen bu bilgilere göre müşteriye kredi verilebilir ya da verilemez kararı ortaya çıkarılmıştır. Bu çalışmada kredi başvurusunda bulunan ve kredi kullanan tüketicilerdeki artışı sağlıklı yönetebilecek yapı ihtiyacının karşılanması, doğru müşteriye, doğru zamanda, doğru miktarda ve doğru vadede kredi vermeyi sağlayacak yapının kurulması, kredi tahsilatlarının verimliliğinin arttırılması, riski minimize ederek karlılığın maksimum noktaya getirecek optimum stratejilerin oluşturulması, bankanın kredi skorlamasında ve değerlendirme sisteminde uzman görüş etkisini azaltılması ve maliyetlerin düşürülmesi amaçlanmıştır.

Anahtar Sözcükler: Kredi Risk Analizi, Kredi Skor Modellemesi, Makine Öğrenmesi, Sınıflandırma Algoritmaları, Gri Kurt Optimizasyonu

Creditability and Classification Work for Finance Sector with Gray Wolf Optimization (GWO) Algorithm

Ali TUNÇ – Kuveyt Türk Participation Bank, R&D Center Konya
Feritpaşa Mah. Kule Caddesi Kule Plaza Kat: 9 No: 1 - 42060 Selçuklu / KONYA
ali.tunc@kuveytturk.com.tr

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Abstract. With the advances in the Information Technology (IT) field, banks can evaluate the credit requests of the customers via effective analytical methods and risk analysis. The software products, named Credit Scoring Systems, consist of collecting customer data based on pre-determined credit factors, processing the data with various statistical or machine learning methods, and conducting credit risk analysis to make the final credit decision. These models utilize the previous transactions on the bank accounts of the customers to make a decision on the credit applications. In the proposed work, the information about the customer related to several aspects and processed with machine learning techniques, and finally a credit score will be determined for each customer. Classification problem using Grey Wolf Optimization method was focused in this work. This information will later be used to decide whether the credit application of a customer can be approved or not. In this study, intentions can be summarized as, providing useful tools to manage the increasing number of customers who apply for consume credits, establishing a structure for crediting the right customers at the right time with the right amount and payment plan, increasing the efficiency of collecting credit payments, thus contributing to the national economy by using the resources more effectively, creating optimal strategies for maximizing the profit by minimizing the risk, reducing the effect of an expert for credit scoring and evaluation, and reducing the costs.

Keywords: Classification Algorithms, Credit Risk Analysis, Credit Scores Modeling, Machine Learning, Gray Wolf Optimization, GWO

Giriş

Kredi skorlama sistemleri; müşterilerin kredi marketine erişim imkânını arttırması, kredi ücretlendirmesinin düşürülmesi, karar süreçlerinin kısaltılması ve karar aşamalarına bağlı ortaya çıkan hata ile temerrütlerin azaltılmasında önemli bir rol oynayarak ülkenin ekonomik gelişimine katkı sağlamaktadır. Yapılan çalışma ile bankaların tüm kredi kararlarını otomatik bir skorlama sistemi üzerinden vermesi, oldukça maliyetli olan banka yetkilisinin kredi sonucunu belirlediği yöntemlere göre daha az maliyetli, hızlı ve kesinliği yüksek bir sistemin oluştura bilirliği incelenmiştir. Yapılan araştırmalar ve istatistiksel verilere göre banka kredilerine duyulan talepteki artışın ve buna bağlı olarak geliştirilen efektif kredi karar modellerinin oluşturulması ihtiyacının somut bir göstergesidir. Bu pazarda başarılı olabilmek için değişen ve gelişen ihtiyaçlara cevap veren efektif, kesinlik oranı yüksek karar mekanizmalarına sahip olmak gerekmektedir.

Bununla birlikte, geliştirilen yeni karar modeli ile elde edilmek istenen en büyük katkı, müşterilere ait yeterli verinin bulunmadığı belirsiz koşullarda ortaya etkin bir sonucun çıkarılmaya çalışılmasıdır. Bu çalışma kapsamında sınıflandırma çalışmaları üzerine yeni geliştirilmiş GWO algoritmasının mevcut problemlerin çözümündeki başarı oranı incelenmeye çalışılmıştır. GWO algoritmasının seçilme nedeni sezgisel olarak sınıflandırma yapan bu algoritmanın finans verilerindeki başarı oranının değerlendirilmesidir. Diğer makine öğrenmesi teknikleri ile ilgili bulunan sonuçlar bu çalışma kapsamında değerlendirilmemiştir. Algoritmaların performans ve başarı oranlarının karşılaştırılması sonraki çalışmalar ile gösterilmeye çalışılacaktır. Bu alanlarda yapılan

geliştirmelerle banka şirketlerinin istatistiksel skorlama modellerinden kaynaklanan kısıtlamalara maruz kalmadan kredi kararlarını verebilmeleri sağlanması hedeflenmiştir. Geliştirilen makine öğrenmesi tabanlı yeni teknolojinin kesinliği yüksek kredi skorlama çözümleri üretilmesinde ciddi faydalar sağlayacağı öngörülmektedir. Çalışmanın genel amacı; kredi kararı aşamalarında oluşabilecek hataları önlemeye yardımcı olan, farklı karar faktörlerinin değerlendirilmesinde standart bir çözüm sunan otomatik skorlama araçları ve modellerinin geliştirilmesidir.

1 Literatür Çalışması

Yapay öğrenme tekniklerinden olan Grey Wolf Optimization (GWO) algoritması hakkında yapılan araştırmalara göre GWO algoritmasının geçmişi 2010 yılına dayanmaktadır. GWO Algoritması için literatür taramasında ulaşılan kaynaklar şunlardır; Mirjalili ve arkadaşları GWO üzerine detaylı bir çalışma yapmışlar ve Parçacık Sürü Optimizasyonu (Particle Swarm Optimization - PSO), Yerçekimi Arama Algoritması (Gravity Search Algorithm - GSA), Diferansiyel Gelişim Algoritması (Differential Evolution Algorithm - DEA) gibi birçok algoritmaya yakın sonuç verdiğini ortaya koymuşlardır. Canis lupus ismi verilen gri kurtların liderlik hiyerarşisi ve av mekanizmasını taklit ederek meta sezgisel bir yaklaşım ortaya çıkarmışlardır. Av ararken, av çevrelerken ve ava saldırırken gösterdikleri yaklaşımı modellemeye çalışmışlardır. 29 tanınmış test fonksiyonu ve test verileri üzerinde benchmarked testlerine tabi tutmuşlar ve PSO, GSA, DEA gibi diğer algoritmalarla rekabetçi sonuçlar ortaya koyduğu bilgisine ulaşmışlardır [1]. Kamboh ve arkadaşları, GWO kullanarak ekonomik yük dağıtım sorununun çözümü ile çalışmışlar, GWO algoritmasını elektrik güç sistemi olmayan dışbükey ve dinamik bir ekonomik yük dağıtım problemi (ELDP) çözümü için kullanmışlardır. GWO algoritması diğer iyi algoritmaları ile karşılaştırıldığında rekabetçi sonuçlar sağlayabildiğini göstermişlerdir [3]. Yusof ve Mustaffa GWO algoritması kullanarak hammadde enerji zaman serisi tahmini üzerine bir çalışma yapmışlardır [4].

Muro, av stratejileri için hesaplama simülasyonlarına basit kurallar ortaya çıkarmıştır [5]. Jayapriya ve Arock, birden fazla moleküler dizileri hizalanması için paralel GWO tekniği kullanarak bir model geliştirmeye çalışmışlar, GWO algoritmasının diğer mevcut olanlara göre hesaplama süresini azaltır olduğunu göstermişlerdir [6]. Mirjalili, çok katmanlı algılayıcıların eğitiminde GWO algoritmasının ne kadar iyileştirici olduğunu göstermeye çalışmış ve Multi Layer Perceptron (MLP) ların eğitiminde GWO yu denemiş elde ettiği sonuçları PSO, ACO, GA, ES ve PBIL ile karşılaştırmıştır. GWO ile elde ettiği değerlerin çok rekabetçi olduğunu ve yakınlaştırmada yüksek değerler elde ettiğini sunmuştur [2]. Korayem, GWO algoritması ile araçların seyahatteki toplam maliyet ve mesafeyi minimize etmek için GWO algoritması kullanılarak bir çalışma yapılmıştır. Elde ettikleri sonuçları ve sorunları diğer algoritmalarla kıyaslamışlar GWO'nun bu problemin çözümünde yakın sonuçlar ortaya koyduğunu göstermeye çalışmıştır [7]. Aaghaee, çok katmanlı algılayıcı için GWO algoritması, bir eğitim algoritması olarak kullanılmış optimizasyon problemini çözmek için geri beslemeli sinir ağlarının eğitiminde GWO algoritmasından yararlanmışlardır [8]. Shakarami ve Davoudkhani, zaman gecikmesi dikkate alarak GWO algoritmasına dayalı geniş alan güç

sistemi dengeleyicisi (WAPSS) tasarımı için bir yöntem önermişlerdir [9]. Sharma ve Saikia, klasik kontrol tabanlı termik güç santrallerinde GWO algoritması kullanarak Multi Area ST'ların otomatik üretim kontrollerini yapmaya çalışmışlardır [10].

2 Gri Kurt Optimizasyonu

GWO algoritması sezgisel optimizasyon yöntemidir. Sezgisel optimizasyon; bir problemin çözümünü, doğruluğunun kanıtlanabilir olup olmadığını önemsenmeden iyiye yakın olarak ele alma yöntemidir. GWO algoritması doğadaki gri kurtların liderlik ve avlanma mekanizmasını taklit eder. Gri kurt hiyerarşisinde Alfa, Beta, Delta ve Omega olmak üzere 4 tip uygulanır [1,2]. Gri kurt hiyerarşisinde lider Alfa kurttur ve sürünün avlanmaya karar vermesi, uyku yeri, kalkma vakti vs. gibi olaylarda sorumluluk sahibidir. Hiyerarşide ikinci sırada bulunan Beta kurt karar verme ve diğer sürü aktivitelerinde Alfa kurdun yardımcısı olarak bilinir. Beta kurt, Alfa kurdun emirlerini diğer kurtlara iletir ve geri dönüş sağlar. Beta kurt, Alfa kurdun yaşlanmasıyla ya da uzaklaşmasıyla onun yerine geçer. Hiyerarşide en düşük seviyeli Omega kurt her zaman yemekte diğer kurtların doymasını ve ona sıra gelmesini bekler. Omega kurdu, her zaman baskın kurt seçer [1,2]. Hiyerarşide üçüncü sırada bulunan Delta kurdu ise eğer kurt Avlanma hiyerarşisinde gri kurtlar öncelikle avlanma davranışına girerler, ardından sırasıyla takip, yaklaşma ve avını izlerler, son olarak takip, rahatsız etme ve çevreleme işlemini yaparlar. Bu izleme, çevreleme ve ava saldırmadır [1].

2.1. Sosyal Hiyerarşi: Matematiksel modellemeyi yaparak GWO tasarımını sağlar. Alfa en iyi çözüm olarak kullanılır. Ardından sırasıyla Beta ve Delta, ikinci ve üçüncü çözüm olarak adlandırılır. Çözüme aday olarak da kalan Omega olarak kabul edilir [1].

2.2. Avı Çevreleme: Yukarıda belirtildiği gibi, gri kurtlar av sırasında avı çevreler. Çevreleme davranışında sırasıyla matematiksel modellemede aşağıdaki denklem 1 ve denklem 2 gibidir [1,2]. Burada t geçerli yinelemeyi gösterir. A ve C vektör katsayısını, Xp avın konumunun vektörü, X bir gri kurdun pozisyon vektörünü gösterir. A ve C Vektörü aşağıdaki denklem 3 ve denklem 4 deki gibi hesaplanır.

$$\vec{D} = |\overrightarrow{C} . \overrightarrow{X_p}(t) - \overrightarrow{X}(t)| \quad (1)$$

$$\vec{X}(t+1) = \overrightarrow{X_p}(t) - \overrightarrow{A} . \overrightarrow{D} \quad (2)$$

$$\vec{A} = 2\vec{a} . \overrightarrow{r_1} - \vec{a} \quad (3)$$

$$\vec{C} = 2 . \overrightarrow{r_2} \quad (4)$$

(x, y) pozisyonunda gri kurt av pozisyonuna göre konumunu günceller (X*, Y*). En iyi ajan A ve C vektör değerlerini güncel konumuna göre değiştirerek farklı yerlere ulaşabilir.Aynı kavram n boyutlara sahip bir arama alanında uzatılabilir. Ve gri kurtlar şimdiye kadar elde edilen en iyi çözüm etrafında hiper küp şeklinde hareket edecektir.

2.3. Avlanma: Gri kurt yerini tanıma ve orayı kuşatma yeteneğine sahiptir. Av genellikle Alfa tarafından yönlendirilir, Beta ve Delta da bazen ava katılabilir. Alfa (en iyi aday çözüm) Beta olduğunu varsayalım ve Delta avın potansiyel konumu hakkında

daha iyi bilgi sahibidir. Bu nedenle, şimdiye kadar elde edilen en iyi ilk üç sonucu kaydedilir ve en iyi arama ajanlarının konumuna göre diğer arama ajanlarının (Omega'lar dâhil) konumu güncellenir. Bu güncellemelere ait denklemler aşağıda denklem 5, denklem 6 ve denklem 7 ile gösterilmektedir [1,2].

$$\overrightarrow{D_\alpha} = \left| \overrightarrow{C_1} . \overrightarrow{X_\alpha} - \vec{X} \right| \overrightarrow{D_\beta} = \left| \overrightarrow{C_2} . \overrightarrow{X_\beta} - \vec{X} \right| \overrightarrow{D_\delta} = \left| \overrightarrow{C_3} . \overrightarrow{X_\delta} - \vec{X} \right| \quad (5)$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \overrightarrow{A_1} . \left(\overrightarrow{D_\alpha}\right), \overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2} . \left(\overrightarrow{D_\beta}\right), \overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3} . \left(\overrightarrow{D_\delta}\right) \quad (6)$$

$$\vec{X}(t+1) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \quad (7)$$

Son konum arama alanındaki Alfa, Beta ve Delta pozisyonlara göre tanımlanan bir daire içinde rasgele bir yerde olacağı görülmektedir. Alfa, Beta, Delta avın konumunu tahmin eder ve diğer kurtlar av etrafında rastgele konumlarını günceller.

2.4. Ava Saldırma (Sömürü): Gri kurt av durduğunda ava saldırarak avı bitirir. Sırayla matematiksel modelle yaklaşan avın a vektörünün değerini azaltıyoruz. Böylelikle a vektörüne bağlı A vektörü de azalır. Başka bir deyişle A vektörü [-2a, 2a] değeri arasında rastgele bir sayıdır ve (a) yineleme boyunca 2'den 0 a düşürülmüştür. A vektörü [-1,1] içinde rastgele değer olduğunda, arama ajanının bir sonraki pozisyonu geçerli pozisyonu ve avın pozisyonu arasında herhangi bir pozisyonda olabilir.

2.5. Av Arama (Keşif): Gri kurtlar kurtların konumuna göre arama yaparlar. Gri kurtlar arama yapmak için dağılır ve avı bulduğunda saldırmak için toplanırlar [1]. Matematiksel olarak sapma modelinde, 1 den büyük rasgele değerler için A vektörü kullanılır. Tanımlanan C vektörü ise [0,2] rasgele değerlerini içerir. Her zaman keşif esnasında rastgele değerleri sağlayabilmek için C gereklidir. Genel olarak özetleyecek olursak, doğada engeller kurtların avlanma durumunda ortaya çıkar, kurdun hızlı ve rahat bir şekilde ava yaklaşmasını engeller, C vektörünün yaptığı da tam olarak budur. Alfa, Beta, Delta ve kurtların ve yırtıcının muhtemel konumunu tahmin edilir. Her aday çözüm yolu için av mesafesi güncellenir. Son olarak, GWO algoritması son bir kriter ile sonlandırılır. Bu algoritmanın sahte kodunu şu şekilde özetleyebiliriz. Gri Kurt Popülasyonunu başlat, Başlangıç değerlerini ata, Herbir temsilci için uygunluk fonksiyon metodu hesapla, X_α en iyi arama temsilcisini ata, X_β en iyi 2. arama temsilcisini ata, X_δ en iyi 3. arama temsilcisini ata, TEKRARLA (HER BİR TEMSİLCİ İÇİ TEKRARLA, Temsilcinin posizyon bilgisini güncelle, Başangıç değerlerini güncelle, Herbir temsilci için uygunluk fonksiyonu metodu hesapla, $X_\alpha - X_\beta - X_\delta$ değerlerini güncelle, t = t+1), Çevrim sayısı maksimum çevrim sayısından küçük olduğu sürece, Geri çevir X_α. Bu işlem sırasına göre algoritma çalışmaktadır.

3 Sonuç

Çalışma kapsamında şu adımlar izlenmiştir. Veri seti oluşturulmuş, oluşturulan veri seti veri ön işleme teknikleri ile temizlenmiş, normalize edilmiş ve kullanılabilir hale getirilmiştir. Sonrasında özellik seçimi algoritmaları kullanılarak veri özelliği

yüksek kullanılabilir bir veri seti haline getirilmiştir. Elde edilen bu veri kümesindeki belirlenmiş özelliklere GWO algoritması uygulanış ve algoritma ile her bir kaydın sonucu kredi verilebilir, kredi verilemez olarak bulunmuştur. Bulunan sonuçlara göre çalışmanın başarı oranları ve doğruluk değerleri hesaplanmış bu hesaplamalara göre de algoritmanın başarısına ait gerekli yorumlamalar yapılmıştır. Bu çalışmada aşağıdaki performans metrikleri kullanılmıştır. Pozitif (P): 'Başarılı' olarak doğru etiketlenmiş müşteri sayısını, Negatif (N): 'Başarısız' olarak doğru etiketlenmiş müşteri sayısını, Doğru Pozitif (TP): 'Başarılı' olarak doğru etiketlenmiş müşterilerin tahmin edilme sayısını, Yanlış Negatif (FN): 'Başarısız' olarak yanlış etiketlenen müşterilerin tahmin edilme sayısını (Gerçekte 'Başarılı' olan müşterinin 'Başarısız' olarak etiketlenme durumu), Yanlış Pozitif (FP): 'Başarılı' olarak yanlış etiketlenen müşterilerin tahmin edilme sayısını (Gerçekte 'Başarısız' olan müşterinin 'Başarılı' olarak etiketlenme durumu), Doğru Negatif (TN): 'Başarısız' olarak doğru etiketlenmiş müşterilerin tahmin edilme sayısını göstermektedir. Burada "Başarılı" terimi "kredi verilebilirliği", "Başarısız" terimi ise "kredi verilemezliği" ifade etmektedir. Aslında problem iki sınıflı bir sınıflandırma problemidir. Performans matrikleri gerçek sonuç ve sınıflandırma sonucuna göre karışıklık Matrisi (Class Confusion Matrix) olarak gösterilmektedir. Matrisdeki sütunlar tahmin sonuçlarına göre, satırlarsa gerçek sonuçlara göre elde edilen sonuçların gösteriminde kullanılmaktadır. Bu değerler literatürde karşılaştırma ölçütü olarak kullanılacak değerlerin üretilmesinde kullanılmaktadır.

Accuracy-Doğruluk: (TP + TN) / (P + N) – Doğru Sonuçların Genel Popülasyona Oranı, Presisyon – Pozitif: TP / (TP + FP) – Kesinlik – Gerçek Pozitiflerin, tüm pozitiflere oranı, Presisyon – Negatif: TN / (TN + FN) – Kesinlik – Gerçek Negatiflerin, tüm negatiflere oranı, Type I Accuracy: FN / (P + N) – Doğruluk – Yanlış Negatiflerin Genel Popülasyona oranı, Bir müşteriyi gerçekte "Başarılı" olmasına rağmen "Başarısız" olarak etiketlemek "Type I Accuracy" terimi ile ifade edilir. 'Başarılı' olmasına rağmen 'Başarısız' olarak ifade edilen bir kaydın finansal açıdan yapılacak çalışmalarda daha kötü sonuçlar doğurabilme olasılığı bulunmaktadır. Bu sebeple başarı oran kriteri olarak "Type I Accuracy" ölçütü tercih edilmiştir.

Bir finans kurumuna kredi talebinde bulunan ve değerlendirilmesi yapılmış 16088 müşteriye ait 157 niteliği olan kayıtlar üzerinde gerekli çalışma gerçekleştirilmiştir. Nitelikler arasında yaş, meslek, evlikik durumu, çocuk sayısı, gelir bilgisi, maaş bilgisi, eş çalışma durumu, araç ve konut bilgileri, toplam kredi bilgileri, ödenen kredi bilgileri, borç bilgileri, ödenmeyen kredi bilgileri gibi kişisel ve finansal veri içeren 157 nitelik bulunmaktadır. 16088 kayıttan oluşan veri kümesinde 13718 "Başarılı" olarak sonuçlanmış, 2370 "Başarısız" olarak sonuçlanmış kredi başvuru bilgisi içermektedir. Çalışılan veri kümesi MS SQL SERVER veri tabanı üzerinde tutulmak için gerekli alanlara uygun tablo tasarımı yapılmıştır. Veri tabanında yer alan alanlar üzerinde veri ön işleme teknikleri kullanılarak aykırı veriler atılmış gürültüler temizlenmiş ve gerekli alanlara normalizasyon işlemleri uygulanmıştır. Veri kaynaklarına erişecek uygulama ara yüzlerin geliştirilmesi C# programlama diliyle geliştirilmiştir. C# ile yazılan GWO algoritması çalıştırılarak sınıflandırma çalışması yapılmış elde edilen 'Başarılı' ve 'Başarısız' kayıt sayılarına ait sonuç değerleri bulunmuştur. GWO algoritmasının seçilme nedeni sezgisel olarak sınıflandırma yapan bu yeni algoritmanın finans verilerindeki başarı oranının değerlendirilmesidir. Eğitim seti 5 parçaya bölünmüştür. K-Katlamalı

Çapraz Doğrulama (K-Fold Cross Validation) yöntemi ile K değeri 5 seçilerek 4 parça (%80) eğitim verisi olarak, 1 parça (%20) ise test verisi olarak kullanılmıştır.

GWO algoritmasının probleme uygulanması şu şekilde açıklanabilir. Her bir özelliğin birbirine ve sonuca olan etkisi olduğu için öncelikle kolon değerleri sonuçlara göre sınıflandırılmıştır. Her bir özelliğin sonuca olan etkisi GWO algoritması yardımı ile hesaplanarak özellik etki değerleri bulunmuştur. Burada her bir niteliği bir kurt indexi olarak yorumlayabiliriz. Bu işlem yapılarak eğitim kümesinde sonucu "Başarılı" olarak kümelenen verilerin her bir özellik etki değeri hesaplanmış ve niteliklerin sonuca etki katsayısı çıkarılmıştır. Aynı işlem eğitim setindeki sonucun "Başarısız" olduğu veriler içinde yapılmıştır. Test edilecek verilerin var olan değerleri sütun etki değerleri ile çarpılarak sonuca ulaşmaya hedefleyen bir uygunluk yöntemi metodu yazılmaya çalışılmıştır. Bu metot sonucunda bulunan değerlere göre "Başarılı" değer "Başarısız" değerden daha büyük bulunursa test verisinin sonucu "Başarılı", daha küçük bulunursa test verisinin sonucu "Başarısız" olarak işaretlenir. Bu işaretleme sonucu gerçek değeri ile bulunan değeri ayrı ayrı kaydedilir. Bu kaydedilen bilgilere göre P, N, TP, FN, FP, TN bilgileri çıkartılır. Bu bilgiler üzerinden Accuracy, Precision-P, Precision-N, Type I Accuracy değerleri hesaplanır. GWO algoritması kullanarak elde edilen sonuçlar aşağıdaki gibi Tablo 1'de sunulmuştur. Bu veriler ile gerekli kazanım hesapları yapılmış elde edilen değerlere göre bulunan doğruluk sonuçları Tablo 2'de sonuç performans karşılaştırma çizelgesinde sunulmuştur.

Tablo 1. GWO Algoritması Sınıflandırma Değerleri

Eğitim Oranı	Test Oranı	KS	P	N	TP	FN	TN	FP
80%	20%	16088	13718	2370	12077	1641	1226	1144
75%	25%	16088	13718	2370	11797	1921	1331	1039

Tablo 2. GWO Sınıflandırma Ölçütlerinin Performans Tablosu

Veri Seti	Performans Metrikleri			
	Accuracy	Precision-P	Precision-N	TypeI Accur.
16088 kayıt	0.8268	0.9134	0.4276	0.1020

Tablodaki değerleri yorumlamak gerekirse veri setindeki alanlar GWO algoritması yardımıyla sınıflandırılmak istenirse 13718 kredi verilebilir kayıtın 12077 tanesi bu algoritma sayesinde de kredi verilebilir olarak bulunmuştur. Kredi verilemez denilen 2370 kaydın 1226 tanesi kredi verilmez olarak bulunmuştur. Bu verilere göre Accuracy değeri hesaplanırsa 0.8268 lik doğruluk oranı ile bu algoritma sınıflandırma işlemini yapabiliyor demektir. Precision-P ve N değerlerini yorumlamak gerekirse P (kredi verilebilir) sonuçlanmış işlemlerdeki başarı oranı N (kredi verilemez) işlemlerdeki başarı oranına göre çok daha yüksek doğrulukta çalışmıştır. Yani veri seti üzerinde kredi verilebilir sonucunda olan kayıt sayısı ve özellik sayısı daha baskın gelmektedir denilebilir. Başarısız (kredi verilemez) olarak değerlendirilmiş kayıt sayısının artırılması ya da veri setine olumsuzluk gerekçesi olabilecek yeni nitelikler, alanların dahil

edilmesi ile bu çalışmanın sınıflandırma sonuçlarının ve doğruluk derecelerinin daha yüksek doğrulukta çalışmasını sağlayabiliriz.

Sınıflandırma algoritmalarında en önemli kısım eğitim aşaması ve doğru modeli ortaya koymaktır. Bu anlamda dikkat edilmesi gereken en öncelikli konu doğru bir veri seti üzerinde sistemi eğitmektir. Çalışmada ortaya konulan en önemli önerilerden birisi yapılacak sınıflandırma çalışmalarında veri setinin gerçekçi veriler içermesini sağlayacak gerekli ön işlemler uygulanmış doğru bir veri setinin kullanılmasıdır. Uygun veri seti sağlandıktan sonra performansı artırmak için sonuca en çok etki gösterecek özelliklerin tespiti için özellik seçim yöntemleri uygulanabilir. Bu adımda başarısı kanıtlanmış algoritmalardan Kazanım Oranı, Bilgi Kazanımı, ID3 Karar Ağaçları, KN komşuluk gibi algoritmalar kullanılabilir. Özellik seçimi algoritmaları yardımıyla bir alt küme üzerinde çalışmak hem hesaplama maliyetlerini azaltacak, hem işlem sürelerini kısaltacak hem de performans artıracaktır. Az sayıda nitelikten oluşan modellerde istatistiksel sınıflandırma algoritmaları, farklı bağlılık değerlerinden ve çok sayıda nitelik barındıran veri kümelerine ait modellerde ise sezgisel algoritmaların daha başarılı sonuçlar ortaya koyduğu yapılan çalışmalar ile gözlemlenmiştir. Testlerde elde edilen modellerde ortaya çıkan sonuca göre GWO yönteminin sınıflandırma çalışmalarında başarılı olduğu tespit edilmiştir. GWO algoritmasının sonuçlarının başarı oranlarının iyi olduğu fakat çalışma süresinin daha yüksek olduğu, geliştirilebilecek metotlar sayesinde sınıflandırma işleminde kullanılabileceği tespit edilmiştir. GWO algoritmasının sınıflandırma işlemi gibi özellik seçimi aşamasında da kullanılabileceği sonucu ortaya çıkarılmıştır. İlerleyen çalışmalarda GWO algoritmasının başarı sonuçları, bayesian, ANN, SWM, SOM gibi algoritmaların sonuçları ile karşılaştırılacak bulunan sonuçlar üzerinde performans ve başarı oranı yorumlamaları yapılacaktır.

Kaynakça

1. Mirjalili, S., Mirjalili, S. M. ve Lewis, A., Grey Wolf Optimizer, Advances in Engineering Software, 69, 46-61, (2014).
2. Mirjalili, S., How effective is the Grey Wolf optimizer in training multi-layer perceptrons, Applied Intelligence, 43 (1), 150-161, (2015).
3. Kamboj, V. K., Bath, S. K. ve Dhillon, J. S., Solution of non-convex economic load dispatch problem using Grey Wolf Optimizer, *Neural Computing & Applications*, 27 (5), 1301-1316, (2016).
4. Yusof, Y. ve Mustaffa, Z., Time Series Forecasting of Energy Commodity using Grey Wolf Optimizer, *Proceedings of the International MultiConference of Engineers and Computer Scientists 2015*, 1, (2015).
5. Muro, C., Escobedo, R., Spector, L. ve Coppinger, R. P., Wolf-pack (Canis lupus) hunting strategies emerge from simple rules in computational simulations, *Behavioural Processes*, 88 (3), 192-197, (2011).
6. Jayapriya, J. ve Arock, M., A Parallel GWO Technique for Aligning Multiple Molecular Sequences, *2015 International Conference on Advances in Computing, Communications and Informatics (Icacci)*, 210-215, (2015).
7. Korayem, L., Khorsid, M. ve Kassem, S. S., Using Grey Wolf Algorithm to Solve the Capacitated Vehicle Routing Problem, *3rd International Conference on Manufacturing, Optimization, Industrial and Material Engineering (Moime 2015)*, 83, (2015).

8. Aaghaee, N., Hayati, M. ve Valian, E., Feedforward neural network training using Grey Wolf Optimizer, *National Conference on Technology*, Energy and the Electrical and Computer Engineering Approach, (2014).
9. Shakarami, M. R. ve Davoudkhani, I. F., Wide-area power system stabilizer design based on Grey Wolf Optimization algorithm considering the time delay, *Electric Power Systems Research*, 133, 149-159, (2016).
10. Sharma, Y. ve Saikia, L. C., Automatic generation control of a multi-area ST - Thermal power system using Grey Wolf Optimizer algorithm based classical controllers, *International Journal of Electrical Power & Energy Systems*, 73, 853-862, (2015).

Makine Öğrenmesi Sistemi Geliştirmek İçin Yazılım Sürecini Uyarlamadaki Zorluklar: Bir Vaka Çalışması

Görkem Giray[1][0000-0002-7023-9469], Murat Osman Ünalır[2][0000-0003-4531-0566], Sena Koçer[1]

[1] Kokteyl A.Ş., İstanbul
[2] Ege Üniversitesi, Bilgisayar Mühendisliği Bölümü, İzmir
gorkem.giray@kokteyl.com, murat.osman.unalir@ege.edu.tr, sena.kocer@kokteyl.com

Özet. Makine öğrenmesi alanındaki gelişmeler yazılım mühendisliğini derinden etkilemektedir. Makine öğrenmesi algoritmalarının mevcut yazılım sistemlerine önemli yetenekler kazandırabileceği çok sayıda örnekte görülmüştür. Bundan dolayı birçok yazılım geliştirme takımı mevcut yazılım sistemlerine makine öğrenmesi yetenekleri kazandırmak için projeler başlatmıştır. Bu çalışmada Kokteyl şirketinin makine öğrenmesi algoritmaları kullanarak tahmin yapabilen bir bileşen geliştirme projesinde yaşadığı deneyimler anlatılmaktadır. Bu deneyimler süreç, veri ve yetkinlikler başlıkları altında incelenmiştir.

Anahtar Kelimeler: Makine Öğrenmesi, Yazılım Mühendisliği, Yazılım Geliştirme.

Challenges in Adapting Software Process for Developing a Machine Learning System: A Case Study

Görkem Giray[1][0000-0002-7023-9469], Murat Osman Ünalır[2][0000-0003-4531-0566], Sena Koçer[1]

[1] Kokteyl Corporation, Istanbul
[2] Ege University, Computer Engineering Department, Izmir
gorkem.giray@kokteyl.com, murat.osman.unalir@ege.edu.tr, sena.kocer@kokteyl.com

Abstract. Developments in the field of machine learning have a profound impact on software engineering. It has been seen in many examples that machine learning algorithms can provide significant capabilities to existing software systems. Therefore, many software development teams have initiated projects to provide machine learning capabilities to existing software systems. This study describes the experiences of Kokteyl Company in a component development project that can make predictions using machine learning algorithms. These experiences are examined under the titles of process, data and competencies.

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Keywords: Machine Learning, Software Engineering, Software Development.

1 Giriş

Büyük verinin işlenebiliyor olması, ucuz veri depolama imkanlarının, esnek hesaplama platformlarının ve algoritmik gelişmelerin bir araya gelmesi, çok sayıda makine öğrenmesi uygulamasının hayatımıza girmesine yol açmıştır [1], [2]. Bu durum, daha önce görece birbirinden bağımsız gelişen yazılım mühendisliği ve makine öğrenmesi (ya da daha genel olarak yapay zeka) alanlarının birbirine yaklaşmasını zorunlu kılmıştır. Bu iki alanın yakınlaşması için "Software Engineering for Machine Learning Applications (SEMLA)" gibi organizasyonlar düzenlenmeye başlanmıştır; hem akademik çevrelerden hem de endüstriden uzmanların takip ettiği IEEE Software gibi dergiler yapay zeka ve yazılım mühendisliği alanlarının kesişiminde neler yapılabileceği konusunda çalışmalarını paylaşmaları için araştırmacılara çağrıda bulunmuştur (IEEE Software dergisinin 2019 yılındaki bildiri çağrısı: The AI Effect: Working at the Intersection of AI and Software Engineering).

Makine öğrenmesi alanındaki gelişmeler ile paralel olarak Kokteyl şirketi de mevcut reklam aracısı yazılımına (RAY) makine öğrenmesi yetenekleri kazandıracak bileşenler geliştirmek amacıyla bir araştırma ve geliştirme projesi başlatmıştır. Bu bildiride, proje kapsamında makine öğrenmesi yeteneklerine sahip bileşenler geliştirmek için şirketin mevcut yazılım mühendisliği sürecinin nasıl değiştirildiği ve bu değişiklikleri yaparken karşılaşılan zorluklar anlatılmaktadır.

2 İş Probleminin Tanımı

Bir reklam aracısı, bir uygulamaya, yani yayıncıya, reklam sağlarken, hangi reklam ağlarından hangi sırada reklam talep edeceğine dair bir karar verir. Bu kararı verirken uygulamanın reklam gelirini (aynı zamanda da kendi reklam gelirini) azami seviyeye çıkarmayı hedefler. Bunun için de reklam gösteriminden en fazla gelir sağlanabilecek reklam ağından başlayarak reklam talebinde bulunur. Mobil reklam sektöründe reklam gösterimi başına geliri ifade etmek için eBGBM (Etkin Bin Gösterim Başına Maliyet; İngilizcesi: eCPM – Effective Cost per Mille) kavramı kullanılmaktadır. Mevcut durumda Kokteyl şirketinin Reklam Aracısı Yazılımında (RAY) gerçekleşecek eBGBM değerleri alan uzmanları tarafından sezgiye dayalı olarak tahmin edilmektedir. eBGBM değerini etkileyebilecek değişkenler tespit edilerek, bu değişkenlerle eBGBM arasındaki örüntülerin makine öğrenmesi algoritmalarıyla tespit edilmesi mümkün görünmektedir [3–6]. Dolayısıyla mevcut RAY'ye makine öğrenmesi yetenekleri kazandırılarak eBGBM tahmini yapan bir bileşen eklemek için bir proje başlatılmıştır.

Gerçekleşen eBGBM değerleri ve bağlantılı veriler RAY'de Tablo 1'de gösterilen alanlarda saklanmaktadır. Her yayıncı uygulamanın her bir reklam alanı için ülke bazında günlük olarak gerçekleşen ve daha önce tahmin edilen eBGBM değerleri RAY tarafından kaydedilmektedir. Her ülkede mobil uygulamaların kullanım dinamiklerinin farklı olması nedeniyle eBGBM değerleri ülke bazında ayrı ayrı tahmin

edilmektedir. Bunun yanında reklam ağları da her yayıncı uygulama için ülke bazında gerçekleşen eBGBM değerlerini reklam aracılarıyla paylaşmaktadır.

Tablo 1. eBGBM değerlerinin ve bağlantılı verilerin saklandığı alanlar ve açıklamaları

Alan adı (Türkçe)	Alan adı (İngilizce)	Açıklama
Tarih	date	tarih
Reklam alanı no	placement_id	yayıncı uygulamanın reklam gösterilebilecek bir konumunun RAY'deki kimlik numarası
Yayıncı uygulamanın adı	app_name	yayıncı uygulamanın adı
Ülke	country	yayıncı uygulamanın kullanıldığı ülke
Gerçekleşen eBGBM	real_ecpm	gerçekleşen eBGBM değeri
Tahmin edilen eBGBM	weighted_ecpm	alan uzmanları tarafından tahmin edilen eBGBM değeri

Örnek olarak bir yayıncı uygulama için (uygulamanın adı gizlilik nedeniyle verilmemiştir) üç günde, dört ülkede gerçekleşen ve daha önce tahmin edilen eBGBM değerleri Tablo 2'de gösterilmektedir.

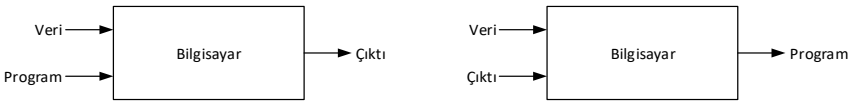
Tablo 2. Örnek eBGBM değerleri ve bağlantılı veriler

Tarih	Reklam alanı no	Uygulamanın adı	Gerçekleşen eBGBM	Tahmin edilen eBGBM	Ülke
08-09-18	00a78a52-7572-4ab6-863c-df5eb77ebe99	yayıncı uygulama	1,280701754	0,012145187	BR
09-09-18	00a78a52-7572-4ab6-863c-df5eb77ebe99	yayıncı uygulama	0,482967144	0,240236608	BR
10-09-18	00a78a52-7572-4ab6-863c-df5eb77ebe99	yayıncı uygulama	0,769817073	0,323807174	BR
08-09-18	00a78a52-7572-4ab6-863c-df5eb77ebe99	yayıncı uygulama	5,929345644	0,026019300	DE
09-09-18	00a78a52-7572-4ab6-863c-df5eb77ebe99	yayıncı uygulama	1,026119403	0,429765486	DE
10-09-18	00a78a52-7572-4ab6-863c-df5eb77ebe99	yayıncı uygulama	1,941533370	0,460431981	DE
08-09-18	00a78a52-7572-4ab6-863c-df5eb77ebe99	yayıncı uygulama	6,718092567	0,021978022	FR
09-09-18	00a78a52-7572-4ab6-863c-df5eb77ebe99	yayıncı uygulama	0,885636856	0,549004147	FR
10-09-18	00a78a52-7572-4ab6-863c-df5eb77ebe99	yayıncı uygulama	2,264134582	0,626127083	FR
08-09-18	00a78a52-7572-4ab6-863c-df5eb77ebe99	yayıncı uygulama	1,640059765	0,005023753	TR
09-09-18	00a78a52-7572-4ab6-863c-df5eb77ebe99	yayıncı uygulama	0,515668347	0,561521377	TR
10-09-18	00a78a52-7572-4ab6-863c-df5eb77ebe99	yayıncı uygulama	0,684581689	0,535929221	TR

eBGBM tahmini yapan bileşenin görevi reklam alanı no, uygulama, ülke gibi özelliklerden yola çıkarak gelecekte gerçekleşecek eBGBM için en iyi tahmini yapmaktır. Bu bileşen bu tahminleri ne kadar iyi yaparsa hem yayıncı uygulamaların hem de RAY'nin elde ettiği geliri artacaktır.

3 Yazılım Mühendisliği ve Makine Öğrenmesi

Geleneksel yazılım geliştirme ile makine öğrenmesi sistemlerinin geliştirilmesi arasında temel bir fark bulunmaktadır. Pedro Domingos bu farkı Şekil 1'deki gibi göstermektedir [7]. Geleneksel yazılım geliştirmede, geliştirilecek sistem ile ilgili gereksinimler müşteri tarafından net olarak tanımlanabilmekte ve bu gereksinimler yazılım geliştiriciler tarafından bir program olarak geliştirilebilmektedir. Bilgisayara veri ve geliştirilen program girdi olarak verilerek beklenen çıktılar elde edilmektedir. Bu süreç deterministik çıktılar üretmektedir. Diğer taraftan bazı problemler için (örneğin nesne tanıma) elde edilecek çıktının nasıl elde edileceği müşteri tarafından tanımlanamayacak kadar karmaşıktır. Bu tür problemlerin bazıları yüksek hacimli veri kümelerindeki saklı desenleri makine öğrenmesi algoritmaları yardımıyla bularak çözülebilmektedir. Bir makine öğrenmesi sisteminin geliştirilmesi sürecinde bilgisayara veri ve elde edilmek istenen çıktı örnekleri girdi olarak verilerek bilgisayarın bir model (aşağıdaki şekilde program olarak adlandırılmıştır) üretmesi beklenmektedir. Aslında bu model, veri kümesi ve çıktı arasındaki bağlantıları, örüntüleri keşfetmekte ve böylece sisteme verilecek yeni veri için modeli kullanarak çıktı için iyi bir tahmin yapmaya çalışmaktadır. Böyle bir sistem için başarımın her zaman yüzde yüz olmasını beklemek mümkün değildir [8].



Şekil 1. Geleneksel yazılım geliştirme (solda) ve makine öğrenmesi sistemi geliştirme (sağda).

Geleneksel yazılım geliştirme ve makine öğrenmesi sistemlerinin geliştirilmesi arasındaki bu temel farktan dolayı çeşitli problemler ortaya çıkmaktadır. Özellikle makine öğrenmesi yeteneklerinin sağladığı faydalar yazılım endüstrisinde çalışan birçok şirketin ilgisini çekmiştir. Bunun sonucunda geleneksel yazılım geliştiren bazı takımlar makine öğrenmesi yetenekleri barındıran bileşenler geliştirmeye başlamıştır. Bu proje kapsamında da böyle bir durum söz konusudur ve bundan dolayı edinilen deneyimler ve yaşanan zorluklar bir sonraki bölümde anlatılmaktadır.

4 Makine Öğrenmesi Sistemi Geliştirme ile İlgili Zorluklar

Proje kapsamında karşılaşılan zorluklar üç başlık altında incelenmiştir: (1) süreç, (2) veri, (3) yetkinlikler ile ilgili zorluklar.

4.1 Süreçle İlgili Zorluklar

Yazılım geliştirme sürecindeki etkinlikler gereksinim mühendisliği, analiz ve tasarım, geliştirme, doğrulama ve geçerleme temel başlıkları altında incelenebilir. Bir yazılım

sistemi kullanılmaya başlandıktan sonra da bakım ile ilgili etkinlikler devreye girmektedir. Bu etkinliklerin nasıl gerçekleştirilmesi gerektiğini belirten, bu etkinlikler arasındaki ilişkileri tanımlayan çok sayıda yazılım geliştirme süreç modeli önerilmiştir. Her şirketin içinde bulunduğu koşullar farklı olduğu için şirketlerin kendi bağlamlarına uygun bir yazılım geliştirme süreci tasarlamaları gerekmektedir [9]. Kokteyl şirketinin mevcut yazılım geliştirme süreci [10] çalışmasında anlatılmaktadır.

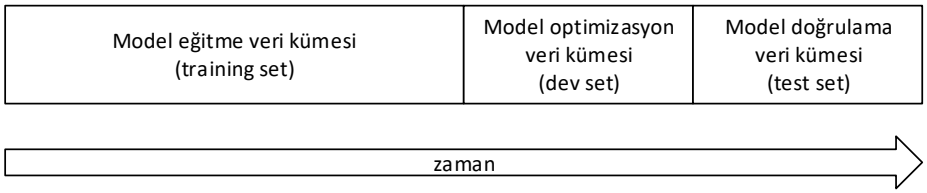
RAY için eBGBM tahminini makine öğrenmesi algoritmaları kullanarak yapan bir bileşenin geliştirilmesi sürecinde mevcut süreçte çeşitli uyarlamalar yapılmak zorunda kalınmıştır. Bunlardan ilki gereksinimlerin belirlenmesi adımında olmuştur. Geleneksel bir yazılımın geliştirilmesi sürecinde gereksinimlerin net biçimde ortaya konulması beklenmektedir [11]. Ancak makine öğrenmesi algoritması kullanan bir bileşeni geliştirme sürecinde gereksinimlerin her açıdan net olarak tanımlanabilmesi mümkün değildir. Bunun nedeni algoritmaların eldeki veriye ve birçok parametreye göre farklı davranışlar sergilemesidir. Bu proje kapsamında üst seviye gereksinim olarak eBGBM değerinin mevcut tahmin bileşenine göre daha başarılı tahminler yapan bir bileşen geliştirilmesi olarak belirlenmiştir. Bileşenin geliştirilmesi sonrasında geçerleme için kök ortalama kare hatası metriğinin kullanılmasına karar verilmiştir [12].

Gereksinimlerin sahip olması istenen önemli özelliklerinden birisi de gereksinimin ne kadar sürede gerçeklenebileceğinin tahmin edilebilir olmasıdır [13]. Ancak makine öğrenmesi algoritmalarının bir problemi çözmek için kullanılması sürecinde çok sayıda belirsizlik bulunmaktadır. Problemi çözme süreci belirsiz sayıda yineleme adımından oluşmaktadır ve sonuçların tahmin edilmesi çok zordur.

Gereksinimlerin belirlenmesi sürecindeki bir başka zorluk gereksinimlerin belgelenmesi konusunda yaşanmıştır. Her ne kadar şirket çevik yöntemlerin önerdiği pratikleri uygulasa da projenin Tübitak tarafından desteklenmesi nedeniyle belirli bir seviyede belgeleme yapılması ihtiyacı doğmuştur. Gereksinimlerin belgelenmesi için Wiegers'ın önerdiği gereksinim belgeleme şablonu [11] kullanılmıştır. Bu şablondaki başlıklar geleneksel yazılım gereksinimleri için yeterli olurken makine öğrenmesi ile ilgili işlevler için şablon üzerinde değişiklikler yapılması gerekmiştir.

Bir başka zorluk ise doğrulama ve geçerleme aşamasında yaşanmıştır. Makine öğrenmesi uygulamalarının test edilmesi için geleneksel test tekniklerinin yeterli olmadığı ve bu konuda başka tekniklerin geliştirilmesi gerektiği araştırmacılar tarafından belirtilmiştir [14]. Doğrulama (geliştirilen işlevin gereksinim belirtimleriyle ne ölçüde uyumlu olduğunun kontrol edilmesi) için eldeki veri üç kümeye ayrılmıştır [8]. Birinci küme makine öğrenmesi algoritmalarının çalıştırılarak bir model elde edilmesi için kullanılmıştır. İkinci veri kümesi elde edilen modelin parametrelerinin optimize edilmesi ve modele özellik ekleme/çıkarma işlemlerinin yapılması için kullanılmıştır. Üçüncü veri kümesi ise elde edilen modelin başarımının ölçümü için yani doğrulama için kullanılmıştır. Şekil 2'de model geliştirme ve doğrulama için kullanılan üç veri kümesi gösterilmektedir. Bu çalışmadaki önemli hususlardan birisi hazırlanan veri kümelerinin zaman boyutunun eldeki problemin çözümü için önemidir. eBGBM tahmini zamana bağlı bir problem olduğu ve geçmişte oluşan değerler ve çeşitli parametreler kullanılarak gelecekteki eBGBM değerleri tahmin edilmeye çalışıldığı için oluşturulan veri kümelerinin zaman aralıkları buna uygun ayarlanmıştır. Model doğrulama veri kümesindeki verilerin zaman aralığı model eğitme ve optimizasyon veri kümesi-

nin zaman aralığını takip etmektedir. Böylece gerçekte modelin gelecek için yapacağı tahmin işlemi simüle edilmiştir.



Şekil 2. Model doğrulama için kullanılan üç veri kümesi.

Geleneksel yazılım sistemleri için geçerleme (geliştirilen işlevin kullanıcı gereksinimleriyle ne ölçüde uyumlu olduğunun kontrol edilmesi) işlemi genellikle kullanıcı kabul testleri ile yapılmaktadır. Ancak bu proje kapsamındaki eBGBM tahmin probleminin çözümü için geliştirilen bileşenin canlı ortamdaki başarımının gözlemlenmesi gerekmektedir. Geliştirilen bileşenin davranışı deterministik olmadığı için geleneksel yazılım sistemleri için yapılan kullanıcı kabul testi bu durumda aynı faydayı sağlamamaktadır. Bunun için bu proje kapsamında eBGBM tahminini yapan bileşenin canlı ortamda yapılan tahminlerin belirli bir yüzdesini yapacak şekilde kullanıma açılmasına karar verilmiştir.

4.2 Veriyle İlgili Zorluklar

Geleneksel yazılım sistemleri için de veri önemli bir yere sahiptir. Yazılım geliştirici tarafından belirlenen ve kodlanan kurallar veri üzerinde işletilmektedir. Diğer taraftan, makine öğrenmesi bileşenleri için veri daha da belirleyici öneme sahiptir [15]. Bunun nedeni makine öğrenmesi uygulamalarında oluşturulan modellerin makine öğrenmesi algoritmalarından çok o modeli eğitmek ve optimize etmek için kullanılan veri kümesine bağlı olmasıdır [16]. Bazı genel problemler için genel amaçlı veri kümeleri model geliştirmek için kullanılabilir. Örneğin, bir nesne tanıma sistemi geliştirmek için ImageNet veri kümesi [17] kullanılabilir. Bu proje kapsamındaki projede ise eBGBM tahmin etme problemi için kamuya açık bir veri kümesi bulunmamaktadır. Diğer taraftan, şirketin veri deposunda üç yıl öncesine kadar mevcut tahmin algoritması ile tahmin edilen ve gerçekleşen eBGBM değerleri bulunmaktadır. Bu veri kümesi kullanılarak bir model geliştirilmesi ve doğrulanması mümkün olmuştur.

Veri ile ilgili karşılaşılan zorluklardan biri veriyi tanımlayan bir üstveri deposunun olmaması noktasında yaşanmıştır. Modeli geliştirecek takım, verinin ne anlama geldiği ve veri alanları arasındaki ilişkiler hakkındaki bilgileri ancak alan uzmanları ile çalışarak zaman içinde öğrenmiştir. Bu deneyim de veriyi tanımlayan üstverinin önemini makine öğrenmesi uygulamaları için de göstermektedir.

Veri ile ilgili karşılaşılan bir başka zorluk ise model geliştirme sürecinde kullanılan veri kümelerinin versiyonlanmasıdır [18]. Model geliştirme ve doğrulama sürecinde kullanılan veri kümelerinin oluşturulan modellerle bağlantılı olarak uygun biçimde saklanması gerekmektedir. Bu proje kapsamında geliştirilen küçük çaptaki modeller için kullanılan ver kümeleri Excel dosyalarında saklanmaktadır. Ancak bu durum

sürdürülebilir değildir. Yüksek hacimli veri kümelerini ve bu veri kümeleriyle bağlantılı modelleri saklayacak bir çözümün geliştirilmesi projede henüz çözülemeyen zorluklardan birisidir.

4.3 Yetkinliklerle İlgili Zorluklar

Makine öğrenmesi algoritmalarının kullanıldığı bileşenlerin geliştirilmesi için birtakım yetkinlikler gerekmektedir. Makine öğrenmesi algoritmaları kütüphaneler şeklinde geliştiricilerin kullanımına hazır olarak sunulsa da bu kütüphaneleri kullanmak için çeşitli yetkinliklere sahip olmak gerekmektedir. Makine öğrenmesi yeteneklerinin geliştirilmesi için gerekli yetkinlikler genellikle veri analitiği uzmanı, makine öğrenmesi uzmanı, veri bilimci gibi unvanlar altında toplanmaktadır [1], [18], [19], [20]. Bu roller altında toplanan sorumlulukların ve dolayısıyla sahip olunması gereken yetkinliklerin farklılaştığı görülmektedir [19], [20]. Bu proje kapsamında, bir veri analitiği uzmanı görev almaktadır. Veri analitiği uzmanı, temel istatistik ve matematik bilgisinin [21] yanında, makine öğrenmesi kütüphaneleri ve model geliştirmek için kullanılabilecek programlama dilleri (Python, R gibi) hakkında eğitim almıştır. Bunun yanında konu alanı uzmanları ile yakın çalışarak mobil reklamcılık sektörü ve bu sektörde oluşan verinin ve bilginin yorumlanması konusunda deneyim kazanmaktadır.

5 Sonuçlar

Bu çalışmada Kokteyl şirketinin makine öğrenmesi algoritmaları kullanarak gelecekte oluşacak eBGBM değerlerini tahmin eden bir bileşenin geliştirme sürecinde elde edilen deneyimler anlatılmaktadır. Gereksinimlerin belirlenmesi ve geliştirilen makine öğrenmesi bileşeninin geçerlenmesi adımlarında geleneksel yazılım geliştirme sürecine göre farklılıklar gözlemlenmiştir. Model geliştirilirken kullanılan veri kümelerinin modellerle bağlantılı olarak saklanması bir başka zorluk olarak karşımıza çıkmaktadır. Ayrıca, geleneksel yazılım geliştirme kültürüne sahip bir takım içerisinde makine öğrenmesi ile ilgili yetkinliklerin geliştirilmesi bir başka zorluk olarak karşımıza çıkmaktadır.

Teşekkür

Bu çalışma TÜBİTAK'ın desteğiyle 3171053 numaralı proje kapsamında yapılmıştır.

Kaynakça

1. Pruss, L.: Infrastructure 3.0: Building blocks for the AI revolution. (2017). https://venturebeat.com/2017/11/28/infrastructure-3-0-building-blocks-for-the-airevolution/, son erişim tarihi: 23 Haziran 2019.
2. Ozkaya, I. (2019). The Golden Age of Software Engineering [From the Editor]. IEEE Software, (1), 4-10.

3. E. Llach, “System for automatically selling and purchasing highly targeted and dynamic advertising impressions using a mixture of price metrics,” U.S. Patent Application No. 10/767,050, 2004.
4. B. O’Kelley, “Method and system for pricing electronic advertisements,” U.S. Patent Application No. 11/006,121, 2006.
5. A. Paunikar and M. Hochberg, “Dynamic pricing for content presentations,” U.S. Patent No. 8,706,547. 22, 2014.
6. O. Chapelle, E. Manavoglu, R. Rosales, “Simple and scalable response prediction for display advertising,” ACM Transactions on Intelligent Systems and Technology (TIST), vol. 5(4), Article 61, 2015.
7. Domingos, P. (2015). The master algorithm: How the quest for the ultimate learning machine will remake our world. Basic Books.
8. Ng, A. (2017). Machine learning yearning. URL: http://www. mlyearning. org/(96).
9. Clarke, P., O’Connor, R.V.: The situational factors that affect the software development process: towards a comprehensive reference framework. J. Inf. Softw. Technol. 54(5), 433– 447 (2012).
10. Giray, G., Yilmaz, M., O’Connor, R. V., & Clarke, P. M. (2018, September). The Impact of Situational Context on Software Process: A Case Study of a Very Small-Sized Company in the Online Advertising Domain. In European Conference on Software Process Improvement (pp. 28-39). Springer, Cham.
11. Wiegers, K., & Beatty, J. (2013). Software requirements. Pearson Education.
12. Giray, G., Ünalır, M.O., & Tahmaz, Ş. (2019). Zaman Serisi Analizi için Hiper-Parametre Seçiminde Grid Arama Kullanımı: Reklam Aracısı Yazılımı ile Bir Örnek Durum Senaryosu. International Data Science & Engineering Symposium – IDSES’19.
13. Essential Scrum: A Practical Guide to the Most Popular Agile Process, 1stEd., Kenneth S. Rubin, Addison-Wesley Professional, 2012.
14. Khomh, F., Adams, B., Cheng, J., Fokaefs, M., & Antoniol, G. (2018). Software Engineering for Machine-Learning Applications: The Road Ahead. IEEE Software, 35(5), 81-84.
15. Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B. & Zimmermann, T. (2019, May). Software engineering for machine learning: a case study. In Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice (pp. 291-300). IEEE Press.
16. Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., ... & Dennison, D. (2015). Hidden technical debt in machine learning systems. In Advances in neural information processing systems (pp. 2503-2511).
17. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). IEEE.
18. Simard, P. Y., Amershi, S., Chickering, D. M., Pelton, A. E., Ghorashi, S., Meek, C., Ramos, G., Suh, J., Verwey, J., Wang, M., & Wernsing, J. (2017). Machine teaching: A new paradigm for building machine learning systems. arXiv preprint arXiv:1707.06742.
19. Kim, M., Zimmermann, T., DeLine, R., Begel, A.: The emerging role of data scientists on software development teams. In: Proceedings of the 38th International Conference on Software Engineering, pp. 96-107. ACM (2016).
20. Kim, M., Zimmermann, T., DeLine, R., Begel, A.: Data Scientists in Software Teams: State of the Art and Challenges. IEEE Transactions on Software Engineering. (2017). doi: 10.1109/TSE.2017.2754374

21. Waller, M. A., Fawcett, S. E.: Data science, predictive analytics, and big data: a revolution that will transform supply chain design and management. Journal of Business Logistics 34(2), pp. 77-84. (2013).

Bir Ölçüm Cihazının Programlama Arayüzü için Model Öğrenme

Atakan Garipler[1], Ege Sorguç[1], Evrim Özel[1], Salih Emre Uslu[1], Mahdi Saeedi Nikoo[1,2], and Halit Oğuztüzün[1]

[1] Orta Doğu Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü, Ankara, Türkiye
{atakan.garipler, sorguc.ege, evrim.ozel, e174168}@metu.edu.tr
oguztuzn@ceng.metu.tr

[2] Spark Kalibrasyon Hizmetleri, Ankara, Türkiye
mahdi_saeedi@sparkkalibrasyon.com.tr

Özet. Programlanabilir ölçüm cihazlarının kalibrasyonu için geliştirilen betikler, ancak hedef cihaz üzerinde çalıştırılarak test edilebilemekte ve hatalarından arındırılabilmektedir. Ancak cihazlar çok pahalı ve hazır bulunması zor olabilir. Çözüm için sunduğumuz EmulateIt, Angluin'in L* algoritması'nı temel alan, IEEE 488.2 uyumlu programlanabilir ölçüm cihazları üzerinde çalışan bir model öğrenme ve taklit etme sistemidir. EmulateIt, elektronik bir ölçüm cihazının test arayüzünü algoritmik olarak öğrenir. Öğrenilen model bir sonlu durumlu makina, özelde bir Mealy makinası şeklini alır. Makina, cihaz arayüzünün taklit edilmesini sağlar. Böylece kalibrasyon mühendisleri, kalibrasyon betiklerinin ilk testlerini cihaz mevcut olmadan yapabilirler. EmulateIt kullanıcılara cihazda hatalara neden olan patikaları ve durumları görmelerine ve betiklerini uygun şekilde hatadan arındırmalarına yardımcı olacak bir grafik kullanıcı arayüzü (GUI) sunar. Durum uzayını izlenebilir kılmak için komut parametre değerleri basitleştirilmiştir. Bu nedenle, betik testlerinin son aşamasında gerçek cihazın hazır bulunması ihtiyacı devam edecektir.

Anahtar Kelimeler: Kalibrasyon, Angluin'in L* Algoritması, Sonlu Durum Makineleri, Model Öğrenme, Aktif Öğrenme, IEEE 488.2

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Model Learning for the Programming Interface of a Measurement Device

Atakan Garipler[1], Ege Sorguç[1], Evrim Özel[1], Salih Emre Uslu[1], Mahdi Saeedi Nikoo[1,2], and Halit Oğuztüzün[1]

[1] Middle East Technical University, Department of Computer Engineering, Ankara, Türkiye
{atakan.garipler, sorguc.ege, evrim.ozel, e174168}@metu.edu.tr
oguztuzn@ceng.metu.tr

[2] Spark Calibration Services, Ankara, Türkiye
mahdi_saeedi@sparkkalibrasyon.com.tr

Abstract. Calibration scripts for programmable measurement devices can be tested and debugged with target device being available. However, such devices can be quite expensive, thus, their availability can be problematic. We present EmulateIt, a model learning tool that algorithmically learns the command interface of an IEEE 488.2 compliant device and makes it available for test script development. EmulateIt implements Angluin's L* algorithm to build a model of the device interface. The learned model takes the form of a finite state machine, specifically, a Mealy machine. The machine provides the emulation of the device interface. Thus, calibration engineers can perform early testing of their calibration scripts in the absence of the actual device. EmulateIt has a graphical user interface so that engineers can see the paths and states that cause errors on the device and debug their scripts accordingly. To keep the state space tractable, actual command parameter values are simplified. Therefore, the actual device would still be needed in the final phase of testing the scripts.

Keywords: Calibration, Angluin's L* Algorithm, Finite State Machines, Model Learning, Active Learning, IEE 488.2

1 Introduction

In our era, calibration of programmable measuring devices are conducted by means of calibration software. Calibration engineers create command scripts and data files that should be given as input to the calibration software, which executes them on the device that will be calibrated. Execution of command sequences lead to different states of the device under calibration which are examined during the calibration procedure. Using SCPI (Standard Commands for Programmable Instruments) [1], a standard command language, calibration engineers are able to write test scripts for calibration procedures. Although determining SCPI commands related to concerned procedure (i.e. writing command scripts) is straightforward, writing data files, which determine adjustments (states) of the device that are to be tested, can be difficult for calibration engineers, since there are obscure dependencies between argument values of commands due to physical (electronic) restrictions of devices. That is, a command-data(argument) pair can and probably will prohibit some other command-data(argument) pairs and violations of such prohibitions make the device enter into an error state. From the perspective of the technical specification of the device, such prohibitions can be thought of as the means of avoiding harmful states. Since there is an excessive amount of possible states, keeping track of such dependencies by analying the manual of the device is burdensome. Therefore, calibration engineers may not be able to create correct calibration scripts at first try, hence, they have to debug the files they have created. The only way of debugging these files is trying them on the associated device. However, calibration laboratories do not have all kinds of programmable measuring devices available at all times. Therefore, most of the time, calibration engineers either have to go to the place where the device subject to calibration is present and debug their files there, or they have to wait for device to be available. That requirement causes time and money losses, and results in unfavorable working conditions for calibration engineers.

EmulateIt offers a solution to this problem by algorithmically learning the command interfaces of devices in order to create a software model which can play the role of the actual device interface in the debugging phase. Using Angluin's L* algorithm [2], which is a deterministic active learning [3] algorithm, the Model Learner component examines the input/output behavior of the actual device iteratively to construct a Mealy Machine [4] model (of the test interfaces) of the device. By trying to depend on the least amount of information/experiments about/on the device, the Learner part of the algorithm produces and conjectures a Mealy Machine that is indistinguishable from the system under learning (from the device in this context). Once a conjecture is produced, the Teacher part of the algorithm tests its correctness. When the correct conjecture is found (i.e. the conjecture is indistinguishable from the system under learning), the L* algorithm returns the Mealy Machine corresponding to the system under learning (i.e. the measurement device), which can then be used by calibration engineers to play the role of the actual device during the testing and debugging of calibration scripts. This solves the problem of cost and space shortages and enables engineers to easily and efficiently debug calibration scripts from their office in the early phases of testing where the command sequences are important. The later phases

of testing, where the actual measured values are important, still need the actual device.

2 High Level System View

EmulateIt has two modes of operation: Learning mode and Emulation mode. The learning mode is the mode of operation in which the test interfaces of devices are modeled in order to be emulated later. Once a device is modeled, the resulting model can be used many times in emulation mode. Using the EmulateIt in emulation mode, calibration engineers are able to replace the actual device by its Mealy Machine model while debugging their calibration scripts.

2.1 High Level View for the Learning Mode

In learning mode, EmulateIt interacts with the PyVISA[1] Application , which is Python wrapper for VISA[2] shared libraries (Virtual Instrument Software Architecture), by sending SCPI commands and receiving error messages, which will be explained in the architecture section. The PyVISA Application transmits the SCPI commands to the programmable measuring device. It then reads error messages from the output buffer of the device and sends them back to the internal components of EmulateIt. The role of calibration engineers in the learning mode is to provide the SCPI Script input, which determines the test interface to be modeled, into EmulateIt. Finally, developers are able to make improvements or modifications to internal components during its life-time.

2.2 High Level View for the Emulation Mode

The main users of EmulateIt are calibration engineers who need the Device Emulator component (which performs the task of running the Mealy Machine models of devices) in order to emulate actual device while debugging the calibration scripts. They give related SCPI scripts and data files to the Device Emulator and receive error messages if their inputs yield an error state. That interaction occurs in a manner similar to the interaction between calibration engineers and calibration software (Calibration software can only be used as a means of executing commands on the device in the presence of the actual device.).

---

[1] https://pyvisa.readthedocs.io/en/latest/

[2] http://www.ivifoundation.org/specifications/default.aspx

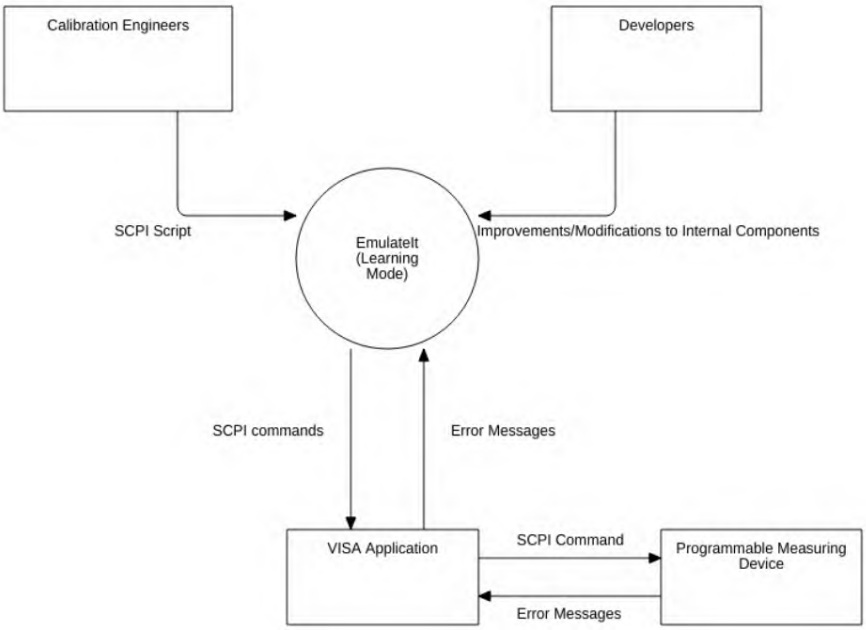


Fig. 1. Context Diagram for the Learning Mode

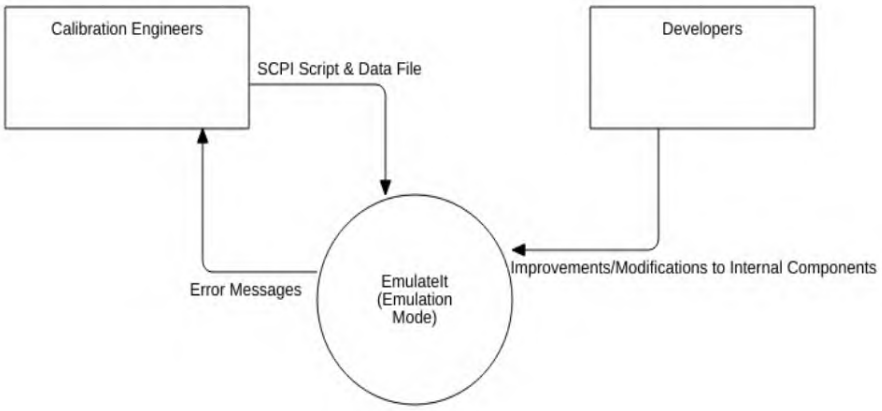


Fig. 2. Context Diagram for the Emulation Mode

3 Architecture

The component diagram of the whole system can be seen in Figure 3. The grey colored components are external components, which are the device that will be modeled and the PyVISA, which is the Python package that enables interaction with programmable measuring devices. The main components of the EmulateIt system are the Model Learner, which fundamentally is the implementation of L* algorithm that creates a Mealy Machine model of the *device under learning (DUL)* by actively doing experiments on it, and the Device Emulator, which is the component to replace the actual device during the debugging phase of calibration scripts. Device Communicator is another component of particular importance due to its role of enabling the Model Learner to functionally and efficiently interact with the *device under learning (DUL)*. Understandably, the Model Learner is the core component of the learning mode and the Device Emulator is the core (and the only) component which is active in the emulation mode. Moreover, all other components are responsible for tasks carried out in the learning mode. The rest of this section is devoted to detailed explanations of the components of the system.

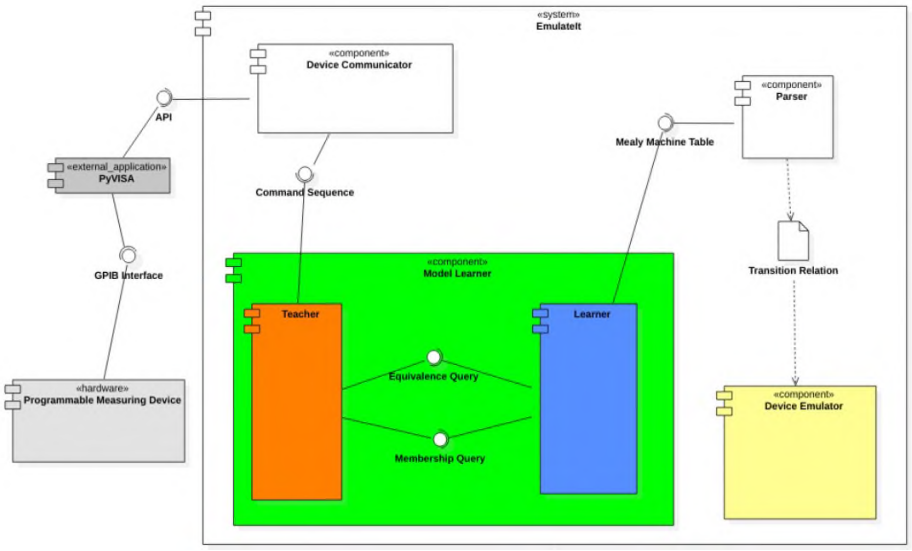


Fig. 3. Component Diagram of EmulateIt

3.1 Model Learner

The Model Learner consists of Teacher and Learner sub-components which correspond to the two parts (with names identical to those modules) of L* algorithm. Teacher and Learner work in conjunction to produce a Mealy Machine table that represents the DUL.

First of all, the L* algorithm has to know the input alphabet symbols of the Mealy Machine (device) under learning. In our design, those input alphabet symbols correspond to concatenations of commands with their possible argument values. Assuming a test interface with n commands each of which can take k argument values is subject to learning, each command is concatenated with k different argument values in order to create alphabet symbols which results in a total of $n \cdot k$ alphabet symbols. To illustrate, assume the (fictional) test interface under learning is constituted of two commands: SOUR:FUNC:SHAP, which sets the waveform of the device and OUTP:LOAD, which adjusts the output load of the device. Assuming the former command can take 3 different argument values: SIN, SQU, TRI (i.e. the device has 3 waveform options) and the latter command can take 2 argument values: 50Ω and 100Ω (i.e. the device has 2 output load options); it can be concluded that the Mealy Machine model of that test interface will have 5 input alphabet symbols: SOUR:FUNC:SHAP SIN, SOUR:FUNC:SHAP SQU, SOUR:FUNC:SHAP TRI and OUTP:LOAD 50, OUTP:LOAD 100. Likewise, error messages (including "No error" message) are considered as output symbols of the Mealy Machine under learning.

Learner interacts with Teacher, which act as the connection between Learner and the operation of the DUL, by sending two types of queries: membership queries and equivalence queries. Membership queries can be thought as experiments done by Learner on the device. They are simply input sequences (of alphabet symbols) which Learner sends, expecting to receive the respective output sequences. Teacher plugs the input sequence into the DUL (into the programmable measuring device in this context, using the Device Communicator) and gets the output (again using the Device Communicator), to send it back to Learner. In fact, for the sake of simplicity Learner can be thought as if it directly sends membership queries to the DUL (and gets outputs directly from the device), bypassing Teacher.

As stated in previous sections of this paper, the L* algorithm maintains an iterative approach in order to infer the DUL. It simply conjectures Mealy Machines (the Learner part) and examines their correctness (the Teacher part). At the very beginning of the execution, Learner assumes a Mealy Machine with only one state and it sends an equivalence query to Teacher. That is, it asks to Teacher whether the assumed (conjectured) Mealy Machine is indistinguishable from the DUL. If Teacher establishes that the two are the same, then it sends back "yes" as output, which means the learning process is completed. If not, Teacher gives a counterexample, which is an input sequence that produces different outputs on the two Mealy Machines (the real one and the one learned until that time). The counterexample is processed (in simple terms, added to the Mealy Machine table which represents the model learned until that point of execution) by Learner and the Mealy Machine structure is recreated by depending on the minimum amount of membership queries, which means conjectur-

ing a new Mealy Machine with more states. The active learning process continues until the DUL (i.e. the test interface of the DUL) is correctly learned.

In order to reply to equivalence queries Teacher creates a set of test sequences, which are (or assumed to be) sufficient to test the correctness of any conjecture. Teacher inputs each test string on both the conjecture and the DUL and compares their outputs. If the outputs given by two machines are different, the corresponding test (input) string is returned to Learner as a counterexample. On the other hand, if none of the test strings produce different outputs on the two machines, Teacher returns "yes" to Learner and the algorithm halts by outputting the learned Mealy Machine.

To sum up, using the two mentioned query types and maintaining an iterative approach, the L* algorithm (the Model Learner) is able to successfully construct the Mealy Machine model of the DUL.

3.2 Device Communicator

In learning mode, the Device Communicator establishes the communication between the DUL and the Model Learner. This communication is crucial during the execution of the L* algorithm, since the algorithm needs to send different types of queries (i.e. input sequences) to the device and receive related outputs in order to learn the device. Particularly, it is the component which communicates with the external PyVISA Application in order to send command sequences produced by the Model Learner to the DUL (in the form of SCPI commands), read error messages from the DUL and then transmit them to the Model Learner, during the learning process.

The other responsibility of the Device Communicator is creating alphabet symbols which will be inputted to the L* algorithm (i.e. to the Model Learner). For that purpose, the Device Communicator includes the dictionary of all SCPI commands and their all argument values. At the very beginning, immediately after receiving the SCPI script which determines the test interface under learning, the Device Communicator matches commands in the given script with the commands in the command dictionary. Moreover, it serves as a pre-processor by detecting unmatched (mistyped etc.) commands. If all commands in the given script are matched with commands in the command dictionary, they are concatenated with each of their possible argument values, thus input alphabet symbols are generated. Generated alphabet symbols are inputted to the Model Learner in order to enable the learning process to be started.

3.3 Parser

The parser acts as the step between the L* algorithm (the Model Learner) and the Device Emulator. The Parser takes the Mealy Machine table that represents the device as input from the L* algorithm, and parses that table to extract the important information (states and transitions) about the DUL, specifically the transition relation of the learned Mealy Machine. Once the table is parsed, information of states and transitions (i.e. the transition relation) is saved as a file which is reachable by the Device Emulator.

3.4 Device Emulator

The Device Emulator is the component with which users (i.e. calibration engineers) interact mostly. It can reach files on which transition relations of the Mealy Machine models of devices are stored and executes them when requested. Once a device is modeled, it can be emulated many times during the life-time of the EmulateIt software. By means of a simple GUI, by giving SCPI scripts and data files as input, calibration engineers can use the Device Emulator instead of the actual device for debugging their calibration scripts. After taking both files as input, the Device Emulator links data combinations in the data file to the commands in the SCPI script line-by-line and applies the resulting command-argument sequences to the Mealy Machine model of the related device. Then, in case of errors, it reports the error-causing sequences. Moreover, it is able to display the visual representations of the learned Mealy Machine models in order to enable the calibration engineers to see the path yielding the error state in the presence of errors.

4 An Example Run

Suppose a calibration engineer has to write calibration scripts for a device, say it is a Type_X device. Suppose now that there is no Type_X device in her laboratory, and she is not able to physically access a Type_X device on which she can test the scripts she has written. However, she has EmulateIt and had previously made it learn a Type_X device. Thus, she can use EmulateIt in emulation mode instead of a real Type_X device in order to test and debug her scripts. Before describing how she will use EmulateIt in emulation mode, let us illustrate how the device had formerly been modeled using EmulateIt in learning mode.

4.1 Learning Mode

To start, by means of GPIB [1] (General Purpose Interface Bus) the device should be connected to the computer on which EmulateIt is installed. Then, EmulateIt should be started in learning mode. On the Learning Mode screen of the GUI, which is shown in Figure 4, the user (e.g the calibration engineer) should arbitrarily entitle the device and the test interface that will be started to be learned in order to be able to access them in the emulation mode. Afterwards, the SCPI Script, which includes commands used in the concerned calibration procedure, is needed to be provided as input in order to identify the test interface. Finally, the user can initiate the learning process by clicking the “Continue” button. When learning process finishes, an information screen notifies the user and the learned model is saved on a file to be used later in the emulation mode. This process can be repeated for each of test interfaces of a device.

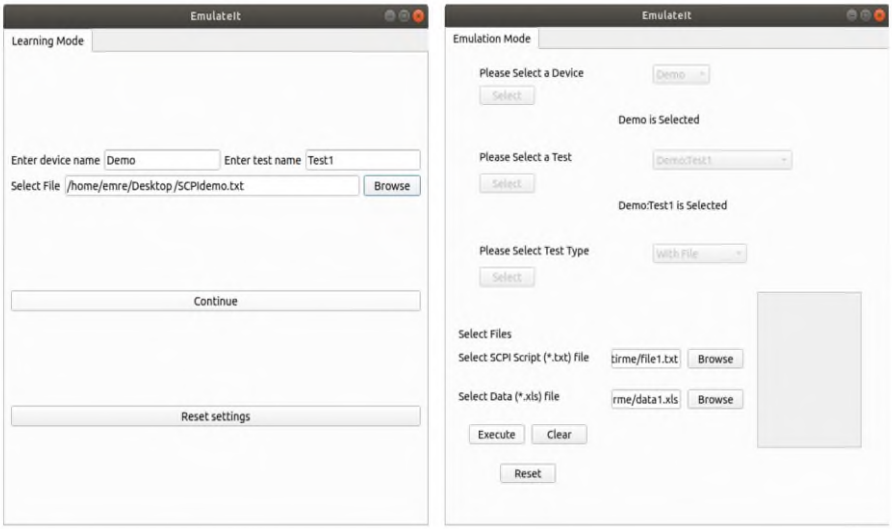


Fig. 4. EmulateIt GUI (Learning Mode on the left, Emulation Mode on the right)

4.2 Emulation Mode

Firstly, user should select Device and Test which refer to devices and tests that are learned before by names given in learning mode by the user. After that selection, the user has two options: Executing the SCPI script and data file on the model of the test interface of the device in the way they are executed on the real device or traversing the Mealy Machine model command by command. When the former option is selected, in case of error, error-causing input sequences are identified and printed on the screen. On the other hand, the latter option draws the visual representation of the Mealy Machine model of the concerned test interface to the screen and visualizes state transitions caused by each entered command. Transitions (commands) resulting in an error are colored red and the path ending with error-causing transitions are colored orange so as to facilitate identification of error-causing dependencies between commands. Thus, this option can be used to simplify the debugging process, especially after the error-causing input sequence has been found.

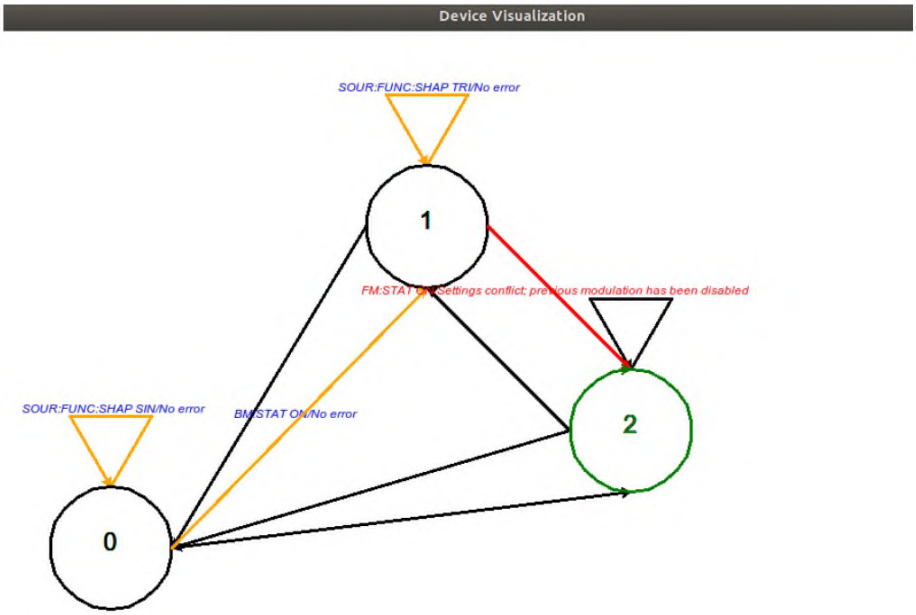


Fig. 5. Device Visualization

5 Discussion

EmulateIt has been tested on Agilent 33120A, its 5 test interfaces, the simplest one of which is constituted of 5 commands and the most complex one of which is constituted of 9 commands, have been learned. Learning durations for these interfaces are measured as being between 56 minutes and 31 hours 43 minutes, which are found to be acceptable.

Although each command-argument value pair has to be an alphabet symbol in order to produce the most precise model as stated in previous sections, commands that can take an excessive amount of different sequential argument values (e.g. SOUR:VOLT, which sets voltage) are not matched with all of their argument values to create alphabet symbols, in order to keep the state space tractable. Instead, margins of their ranges (i.e. min and max values) and a few intermediate values are used. Thus, for those commands, ranges in which errors occur are able to be determined by the Model Learner. Moreover, using ranges instead of exact values does not significantly alter the resulting Mealy Machine structure. The reason for that is, the resulting machine becomes different only if at least two different values in predetermined ranges cause different errors when applied to at least one of the states of the target device and determining reasonable intermediate values greatly reduces the probability of such occurences. Hence, in spite of possible precision loss due to this method, EmulateIt is capable of building highly precise models. Nevertheless, selecting effective intermediate values may not be sufficient in some cases and the real device may be required at the last stage of debugging, in order to remove hard-to-identify bugs.

6 Conclusion

We have shown that a combination of the L* learning algorithm, PyVISA, and some utility modules that transform the data representation (the Parser etc.) is sufficient to construct a Device Emulator, which solves the problem that was initially originated from the need to have physical access to the device. By constructing finite state machine models of test interfaces of devices, EmulateIt provides calibration engineers a much easier way to debug their calibration scripts. With its simple GUI, even engineers that have never seen EmulateIt before can spontaneously use it. Since L* algorithm learns finite state machines algorithmically and deterministically, the final models of test interfaces are concise and of high precision, which increases the level of reliability presented to its users. As stated in the discussion section, some precision loss due to the requirement of keeping the state space tractable may occur. Therefore, at the last phase of debugging, the real device is still needed. However, after debugging them on EmulateIt, remaining bugs on the calibration scripts can be easily and quickly removed by testing the scripts on the real device.

Acknowledgement

This work has been supported by TÜBİTAK TEYDEB under project 7180133.

References

1. IEEE Std. 488.2-1992 – IEEE Standard Codes, Format, Protocols to Use With IEEE Std. 488.1-1987, IEEE Standard Digital Interface for Programmable Instrumentation.
2. Angluin, D.: Learning Regular Sets from Queries and Counterexamples. Information and Computation 75, 87-106 (1987).
3. Vaandrager, F.: Model Learning. Communications of the ACM 60(2), 86-95 (2017).
4. Mealy, G.: A Method for Synthesizing Sequential Circuits. Bell System Technical Journal 34, 1045–1079 (1955).
5. Agilent 33120A Function Generator User's Guide, Agilent Technologies, CA (2002).
6. Agilent 33120A Function Generator Service Guide, Agilent Technologies, CA (2002).
7. Steffen B., Howar F., Merten M.: Introduction to Active Automata Learning from a Practical Perspective. In: Bernardo M., Issarny V. (eds) Formal Methods for Eternal Networked Software Systems. SFM 2011. Lecture Notes in Computer Science, vol. 6659, pp. 256-296. Springer, Berlin, Heidelberg (2011).

Hata Verisi Analizi Ve Değişkenlerin Girdi Uzayının Ayrıklaştırılmasını Kullanan Bir Birleşimsel Test Verisi Üretimi Yaklaşımı

Hakan BOSNALI

ASELSAN, Ankara, Türkiye
Orta Doğu Teknik Üniversitesi, Ankara, Türkiye
hakanbosnali@gmail.com

Özet. Birleşimsel Test yöntemi verimli bir test stratejisidir. Bu yöntem, çoğu hatanın nispeten düşük sayıda değişkenin etkileşimi sonucu meydana geldiği kanısına dayanır. Fakat farklı yazılımlar için doğru etkileşim sayısını belirlemek verimlilik anlamında bir sorundur. Bununla birlikte, girdilerin ayrık bir biçimde olması gerekmektedir. Hata verisi analizi ile doğru etkileşim sayısını belirlemeyi ve değişkenlerin sürekli olan girdi uzayının iyi bilinen test teknikleri kullanılarak ayrıklaştırılmasını bir araya getiren yeni bir birleşimsel test verisi üreteci öneriyoruz. Bu yeni araçla, test maliyetleri en aza indirilirken, test verisine olan güvenin de en üst düzeye çıkarılması hedeflenmiştir. Yeni araçla yapılan deneyler de, sonuçlarda görülen test verimliliğinin önemli ölçüde artış göstermesi ile bu fikri desteklemektedir.

Anahtar Kelimeler: Birleşimsel Test, Hata Verisi Analizi.

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

A Combinatorial Test Data Generation Approach Using Fault Data Analysis And Discretization Of Parameter Input Space

Hakan BOSNALI

ASELSAN, Ankara, Turkey
Middle East Technical University, Ankara, Turkey
hakanbosnali@gmail.com

Abstract. Combinatorial Testing is an efficient testing strategy. It is based on the idea that many faults are caused by interactions between a relatively small number of parameters. However, determining the right interaction strength to generate data for different software is an issue in terms of efficiency. In addition to that, it requires the inputs in a discrete form, while that is not always the case. We propose a new combinatorial test data generator tool that combines fault data analysis to determine the right interaction strength for the specific domain of software and transformation of the continuous input space of parameters into discrete using well known test techniques. With this new tool, it is aimed to minimize test costs, while maximizing the confidence in test data. Experiments made with the tool support this idea with results showing a significant increase in test efficiency.

Keywords: Combinatorial Testing, Fault Data Analysis.

1 Introduction

Generating efficient test data with limited resources has always been an issue for software testing. Most of the time, testing all possible combinations is impossible, since testing becomes exhaustive as the number of inputs increases. Among the numerous available methods, Combinatorial Testing is proven to be an effective testing strategy [1,3]. However, selecting the right degree of interaction is a problem and that affects the efficiency directly. For different software from different domains, the degree of interaction changes [1]. On the other hand, to be able to use this method, the range of inputs must be in a discrete form, but this is not the case at all times. Continuous ranges for numbers and uncountable combination of characters in a string field may not be suitable examples for inputs of this method. For these reasons, before using Combinatorial Testing, degree of interaction for the specific domain of software must be determined and the continuous input fields must be transformed into discrete.

In this paper, we aimed to develop an approach that overcomes these deficiencies. Since our working area is Black-box testing of software from Electronic Warfare and Radar domains, we had a valuable fault data for Graphical User Interface and Embedded software in these domains. Our purpose was analyzing these fault data to obtain the interaction strengths for each domain. Afterwards, we wanted to develop a combinatorial test data generator tool that first transforms the continuous input fields into discrete using well known test techniques, then generates combinatorial test data with the help of an existing tool and using the analysis results for the specified domain to select the degree of interaction. This way, the lacking parts in combinatorial testing method would be completed.

To validate this new tool and the method, we had to design experiments on a real industrial software from one of our domains to see if it helps reducing the effort and increasing the number of bugs found. We also wanted to compare the tests at the same setup with randomly generated test data to create a comparison baseline for the new approach. In these experiments, we tried to find out the duration of preparing test data and running the tests, the number of tests generated and the number of bugs found.

2 Background

Testing all possible combinations of parameters becomes exhaustive as the number of inputs increases. When there is a shortage of resources in cases like this, Combinatorial Testing method is very useful. It is a method that aims to increase the effectiveness of test procedure by reducing the number of test cases by selecting meaningful data according to a strategy. This method makes use of the idea that not every input parameter contributes to every fault generated by the system and many faults are caused by interactions between a relatively small number of parameters [1].

Pairwise Testing, which is a combinatorial method, suggest that many faults in the system are triggered by the interaction of two parameter values. Therefore, while generating input to the system, every combination of values of any two parameters must be covered by at least one test case.

Some empirical investigations have concluded that from 50 to 97 percent of software faults could be identified by pairwise combinatorial testing [1]. Therefore, in case of limited resources, Pairwise Testing may be a good way to find most of the faults in the system.

T-way Testing, which is the generalized form of Pairwise Testing, is based on the idea that some faults may be triggered by the interaction of more than two parameters. To generate these faults, interaction of t parameters is needed. Hence, in T-way testing, all combinations of any t parameters' values must be included in at least one test case.

A 10-project empirical study [1] has shown that some faults were triggered by three, four, five, and six-way interactions in some systems including medical devices, browser, web server and NASA distributed database.

In the light of these studies, T-way testing is better than pairwise testing when more resources are available or more precise testing is required.

3 Related Work

The study [2] investigates the relation between the software fault types and the average t-way interaction that causes them. The terms "Bohrbugs" which are simple bugs and easy to reproduce, and "Mandelbugs" which are complex bugs and more difficult to reproduce are analyzed in terms of average interaction strength.

In this study, 242 bugs from bugs.mysql.com for MySQL software were examined. The results of the analysis on these data have shown that Mandelbugs have higher interaction strength than Bohrbugs. This means that more factors are needed to trigger more complex errors.

It is stated that the curves of two bug types are similar to each other but shifted by one factor. To be able to find all of bugs, 4-way testing for Bohrbugs and 5-way testing for Mandelbugs are sufficient.

To sum up, the idea that T-way testing of up to 6 factors provides the same level of confidence as exhaustive testing turned out to be true in MySQL case.

After analyzing the fault data, it is time to find a method to generate input data for the tests. A combinatorial design based approach for testing pairwise, or t-way combinations of system parameters is proposed in [3]. It takes the system parameters and relations between them as an input. System parameters must be defined in terms of possible values they have. However, these parameters may have some restrictions and constraints that affect the values of each other. For this reason, there is an interface for the user to keep out the disallowed tests. Then, using all these information, a combinatorial algorithm generates the test data according to the interaction strength determined by the user.

Some experiments are conducted on this approach. The results of the experiments with the AETG system demonstrate that it is an effective and robust method to produce input data for the tests.

Phases of Combinatorial Interaction Testing is explained in detail in [4]. Four phases consists of Modeling, Sampling, Test and Analyze. Modeling and Sampling phases define "What" to test, and Test and Analyze phases define "How" to test.

Modeling includes input space of SUT, configurations, constraints of system parameters etc. An input must be stated as a parameter that can have discrete values. If its possible values are continuous, then some common techniques like "Equivalence partitioning" and "Boundary Value Analysis" can be used to transform it to discrete. There may also be some constraints between the input parameters. To avoid invalid combinations of these parameters, constraints must be taken into consideration. Other than these, user may want to define some test data that must be included into test cases, or to remove unwanted or already tested cases. All of these forms the Modeling phase together. The problem with the Modeling is that generation, modification and maintenance of it must be done manually in most of the cases. It should be done in an automatic manner including constructing input space, generating constraints and configuration, and helping the decision of interaction strength.

4 Fault Data Analysis

In this part of the study, we gathered a valuable fault data from a total of 6 projects and 30 Computer Software Configuration Items (CSCI) in the Electronic Warfare and Radar domain. The projects consist of both embedded and Graphical User Interface software. Each project has on the average 3400 requirements and 584000 Lines of Code. In total, there were 1461 faults. We collected 10 features for each fault, which include Detailed explanation of the fault, Detection phase, Fault type, Severity, Reproducibility, Source of the fault, Solution type, Effort for solution, Verification method and Effort for verification.

We analyzed these data in order to design our future roadmap for improving our software test activities and making it more effective. While analyzing, we tried to find answer to certain questions. The question with the most remarkable answer was; What is the number of factors involved in failures?

First we categorized the projects according to their domain; Electronic Warfare (EW) and Radar, and software type; Embedded and GUI. After this categorization, we analyzed the "Detailed explanation of the fault" field to extract the interaction strength information for each fault. About 21% of the faults were not suitable for this analysis due to insufficient information given in the fault description.

Results were in harmony with the idea that 6-way combinatorial testing is effective as exhaustive testing [2]. In Fig. 1, number of interaction between parameters and corresponding cumulative error detection rate for all types of bugs are given. Radar Embedded bugs reach 100% coverage with an interaction strength of 4, Radar GUI bugs with 5, EW Embedded bugs with 5 and EW GUI bugs with 6. Hence, similarly in our case, 6-way combinatorial testing would be enough to find all the bugs.

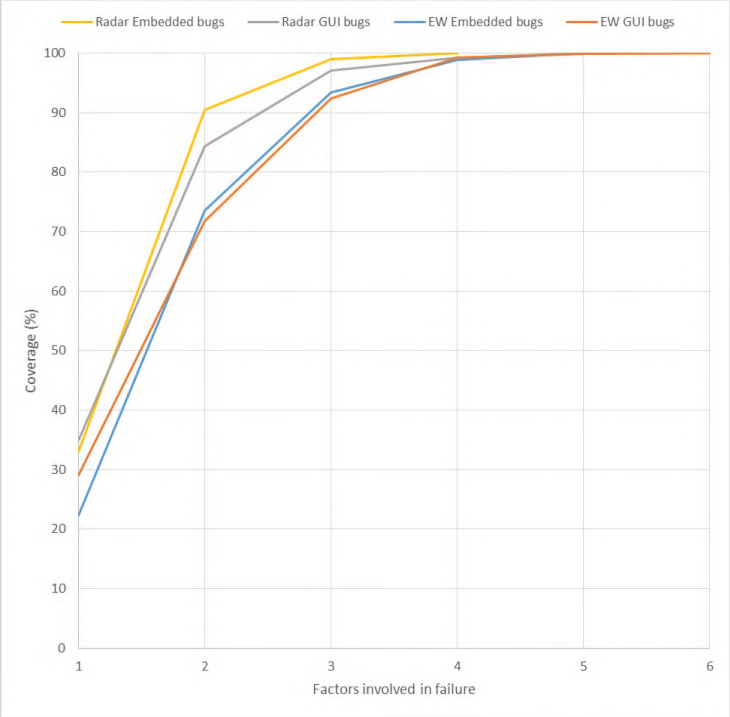


Fig. 1. Interaction strength for EW and Radar bugs

5 Combinatorial Test Data Generator

After fault data analysis, we wanted to find a test data generation method using the results of the analysis. Since we had the interaction strength for each fault type and they were in compliance with the assumption "T-way testing of up to 6 factors provides the same level of confidence as exhaustive testing" made in [2], we decided to use combinatorial testing approach which is known to be an effective testing strategy [1,3].

First we investigated the combinatorial test data generator tools available. Most of the existing tools were built for pairwise testing [6] (e.g. IPO, AllPairs, Jenny). We needed a tool that is capable of generating test data up to 6-way interactions of input parameters according to our analysis results. Moreover, it had to be a desktop application that provides an API for external tools. FireEye t-way testing tool, later named as Advanced Combinatorial Testing System (ACTS), presented in [5] seemed to be suitable for these constraints and performed better than its counterparts [6]. It supports t-way test set generation with t ranging from 1 to 6. It provides a GUI and a command line interface for external usage to generate combinatorial test data with various options. Thanks to these features, it satisfies most of our restrictions.

Although ACTS answers most of our needs, it only accepts inputs in a discrete way. In our systems, we had continuous ranges for some parameters and this needed

to be handled first to be able to use ACTS. Therefore, we needed to develop a tool that completes the lacking abilities of ACTS for our test data development process.

5.1 Design Phase

We designed a tool that takes parameters and their possible values and transforms them into discrete values using well known test techniques. Then, making use of the domain specific interaction strengths that we obtained during the analysis phase, it generates an input file for ACTS tool to generate combinatorial test data.

Our design structure consists of 3 stages: configuring the test data generation, discretizing the parameter values and generating the combinatorial test data using ACTS tool.

Configuring the test data generation. Our tool has two input files to configure the test data generation: one for the domain specific interaction strengths and one for the system parameters.

Interaction strength yields for the value "t" in t-way combinatorial testing. It also refers to "degree of interaction (doi)" between the parameters. In our tool, the interaction strengths for different domains can be configured easily with an xml file. We generated an xml file using the analysis results in Section 4 and added 4 domains and their interaction strength values. While selecting these values, we needed to make a trade-off between the number of test cases and the fault coverage because the number of test cases become larger as the interaction strength increases for full t-way coverage. For that reason, we analyzed the results in Section 4 and came up with the idea that the minimum interaction strengths that cover 90% or more of all faults would be optimum. According to this strategy, we generated the degree of interaction "doi" for each domain. This xml file can easily be extended for different domains just by adding a new line for the new domain and the degree of interaction value.

The second input file is for the system parameters, their values and test design techniques. It also contains the name of the system and the domain which the system belongs to. It can be in both xml and json format.

In the parameters section, for each parameter, there is a parameter name, resolution of its possible values, possible values and test design techniques that will be used in discretizing the continuous values. Possible values field may differ according to parameter type. Users may define a range, enter some specific values that must be included in the test data, and a string with a specified length. If the parameter is a string, then resolution field stands for the length of the string. Otherwise it implies the minimum amount of change inside the defined range. A parameter can have more than one technique to transform its continuous values to discrete.

Some combinations are not valid from the domain semantics, and must be excluded from the resulting test set. There is a constraints section inside the input file that allows users to specify conditions that will be taken into account during test data generation.

Discretizing the parameter values. Once the system, parameters and test techniques are created, the second step is discretizing the parameter values. In this step, we made use of the well known test design techniques such as Equivalence Partitioning and Boundary Value Analysis. We also added Error Guessing and String Analysis for the first version.

EquivalencePartitioning technique discretizes the continuous input by selecting a random value inside the given range of the parameter taking resolution into consideration. BoundaryValueAnalysis technique selects min and max values in the range of the parameter, also (max+resolution) and (min-resolution) values. ErrorGuessing technique just selects the values given in the input file. This technique was added to give the user the ability to add values that must be tested according to their experience. StringAnalysis technique is used to discretize the parameters of string type. It uses the resolution in the input file as the max length of the string, then selects a string with zero length, a random string with the length 1, and a random string with the max length.

After reading the input file, each parameter in the system transforms all continuous and undetermined values into specific, discrete and meaningful data using techniques defined for them.

Generating the combinatorial test data. Once we obtain the discrete parameter values, we need to feed these values to ACTS tool to generate the combinatorial test data. To be able to use ACTS tool, all system specifications, such as parameter values and constraints, should be transformed into an input file format which ACTS tool can understand. For the purpose of using ACTS tool, we generate a temporary input file with the information we have in our system.

After creating the input file for ACTS tool, we need to collect the domain specific degree of interaction value for the System Under Test (SUT). Our system parses the domain.xml file and retrieves the integer degree of interaction value for the domain which SUT belongs to. This value is used as the "t" value for t-way combinatorial test data generation.

Since we have the compatible input file and interaction strength for the parameters, we can run ACTS tool to generate test data. ACTS has a command line interface that helps users to exploit its features without GUI interaction. It takes the degree of interaction, the input file path and the output file path as command line parameters, then generates and writes the test data into the provided output file. We give the generated temporary input file path, degree of interaction value gathered for the SUT, and the output file path where the generated test data will be written.

With this final step, generation of t-way combinatorial test data for our system is completed.

6 An Experiment Using The Test Data Generator Tool

Once constructing the test data generator tool, we designed experiments to evaluate the tool to see if it really helps reducing test data generation effort and increasing the

number of bugs found. For this purpose, we chose a real industrial software in Radar Embedded domain, with a size of 7000 Lines of Code. This software was tested before by generating test data manually. It had more than 180 parameters and therefore many possible cases to be tested. Generating test cases for this software required a huge effort; that is why we chose it for subject program (SUT) in our experiment to see the effect of our tool on test effort.

One problem with the old test setup was setting parameter values manually, which required too much effort while our SUT had more than 180 parameters. Other than the effort of generating test data, another problem was selecting efficient test data with limited resources. In other words, since testing all possible cases was impossible, we needed to select the ones with higher chance to trigger bugs. For these reasons, our test data generator tool seemed to be a perfect fit for this case.

6.1 Test Environment Setup

We have an automatic test infrastructure which was developed in house for the purpose of testing software interfaces and certain test scenarios. It contains the interface descriptions and parameter information between different software. We can generate various test scenarios using this interface descriptions and set different parameter values. Then these scenarios are run automatically and expected test results are compared with the actual ones.

This test automation infrastructure uses an xml file that contains parameter names, their possible values or ranges, test techniques that will be used to analyze them, and constraints between these parameters. This xml file is created automatically from an interface description file that is used for developing the SUT. Then, our infrastructure uses this xml to create a test scenario.

First, we constructed a base test scenario using SUT's interface descriptions. After that, we needed to generate test cases using our test data generator tool. However, the test scenario created by the infrastructure had to be converted to the input format of our tool. We wrote a simple adapter software for this purpose. It works as a translator between the test infrastructure and our test data generator tool to convert each tool's output to the other one's input. This way our test data generation cycle shown in Fig. 2, became fully automatic.

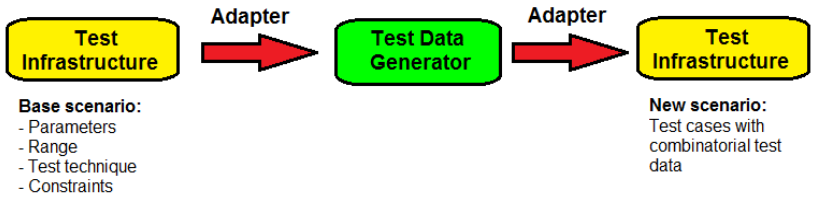


Fig. 2. Test data generation cycle

The adapter software converts the parameter information inside the base scenario into an input file for the test data generator tool. Then the tool generates the combinatorial test data for the selected domain using the related interaction strength which we obtain from the domain.xml file. First, the tool discretizes the possible values of the parameters according to test techniques and then generates the combinatorial test data according to constraints. After generating the test data, the adapter software generates the expected outcomes for input data with the help of interface descriptions and converts these into a test infrastructure scenario file to complete the cycle.

6.2 The Experiment and Test Results

Throughout the experiment, we tried to answer certain questions and obtain some useful metrics to compare the old and the new method. These questions were:

- How long did it take to prepare input files?
- How long did it take to carry out tests?
- How many tests were generated?
- How many bugs were triggered?
- How many new bugs were found?
- How many old bugs went unnoticed?

In the old method, we prepared input data and the expected outcomes manually with the help of our test infrastructure and ran them automatically. The process consisted of preparing one input and the related outcome at a time and running the test. Thanks to automatic test infrastructure, running one test took approximately 1 minute. However, preparing the input data and the expected outcomes took a huge effort. Taking that into account, preparing test cases and running the tests together cost 40 staff-hour in total. 190 test cases were generated manually for the tests and 11 bugs were found.

In the new method, we generated input data and expected outcomes with the test setup explained in Section 6.1. Preparing the base test scenario file took a little longer than the other steps in the test data generation cycle. After that, it only took 6 seconds to create test cases with our new setup. In total, generating test cases and running the tests cost only 8 staff-hour. Our combinatorial test data generator created 45 test cases and 15 bugs were triggered with this data set. Once we analyzed the results, we saw that 4 new bugs were discovered, and no old bugs went unnoticed with this new method.

When we analyzed newly found bugs, we saw that they were all triggered by the interaction between parameters. This showed that combinatorial testing strategy was effective for us to find new bugs. Two of these faults were exceptional cases where they could only be triggered by the interaction of two parameters when each value was at different partitions. Here, equivalence partitioning technique helped us find these bugs. Another fault was triggered by the interaction of two parameters when only one of them was outside the minimum boundary and the absolute value of it was smaller than the other parameter. Boundary value analysis technique helped us trigger

this fault. The last fault was caused when two parameters had string values with the length of zero. In this case, string analysis technique was effective.

The new method reduced the test effort by 80%. Moreover, although the number of test cases decreased by 76%, the number of bugs triggered increased by 36%.

To generate a comparison baseline for the new method, we also designed an experiment with randomly generated test data at the same test setup. For each parameter, we picked a random value in the range of the parameter for each test case and generated 45 test cases as in the new method. The results showed that only 7 faults were discovered where all the faults were caused by the value of one parameter and no interactions were needed to trigger them. When we compare this case with our new method, we see 114% increase in the number of bugs found.

6.3 Threats to Validity

The construct validity is related to possible issues in comparison baseline for our method. In the old method, tester's bias while selecting test cases might have affected the results. To eliminate this threat, we designed another experiment with randomly generated test data in the same test setup. This way, the effect of tester was removed, and we had an objective test data and a more valid comparison baseline.

The internal validity is related to discretization techniques and test data generation algorithm in our approach. Different techniques and tools might affect test data and result in different behaviors. To overcome this threat, we used well known test techniques for the discretization of the parameter input space, and an existing tool (ACTS) to generate combinatorial test data. We selected these after reviewing all available techniques and tools in the literature.

The external validity is related to domain of software that is used in this study. We only applied our approach on Electronic Warfare and Radar domains, thus further analysis on software from different domains should be done for applicability.

The reliability states that this approach can be repeated by other researchers. For this purpose, the steps in Sections 4, 5 and 6 are explained in detail.

7 Conclusion

Software in Electronic Warfare and Radar domains have a large number of parameters with continuous input space. This situation results in uncountable possible combinations for test case generation. Since it is impossible to test all these combinations, we needed to find a method to minimize the number of test cases, while selecting ones with higher possibility of triggering bugs. Combinatorial Testing, which is proven to be an efficient testing strategy, seemed to be a good choice for our situation. However, it required inputs in a discrete form. Moreover, degree of interaction must have been different for each domain of software in terms of test efficiency issues.

The approach we proposed fills these gaps with fault data analysis to determine the right degree of interaction according to domain of software, and discretization of parameter input space using well known test techniques. Fault data analysis that we

made on different software domains helped us in specifying predefined degree of interaction values for each domain. These values were specified for future test data generation of specific domain of software. Discretization of continuous parameter input space was made using test techniques like Equivalence Partitioning, Boundary Value Analysis etc. This way, continuous range for a parameter was reduced to meaningful and discrete data with higher possibility to trigger bugs.

Our approach combined these methods with combinatorial testing to create an efficient test data generation method. Experiments showed that the new method reduced the test effort by 80%, the number of test cases decreased by 76%, and the number of bugs triggered increased by 36% compared to the old method. Moreover, it increased the number of bugs by 114% compared to tests with random data. In addition to these improvements, analysis of the new faults showed that test techniques and combinatorial approach directly played a role in finding them. These results indicates that this new approach is an effective way that minimize test costs, while maximizing the confidence in test data.

This approach was only applied to software from Electronic Warfare and Radar domains. However, since the approach is in a general form, it can be easily applied to software from other domains. This way, it can contribute to efficiency and reduce test data generation costs for software in all domains.

In the future, this approach can be extended to work with other test infrastructures as well. For this purpose, a generic API for external test automation tools can be provided.

References

1. Kuhn, R., Kacker, R., Lei, Y., & Hunter, J. (2009). Combinatorial Software Testing. Computer, 42(7), 94-96.
2. Z. Ratliff, R. Kuhn, R. Kacker, Y. Lei, K. Trivedi, "The Relationship Between Software Bug Type and Number of Factors Involved in Failures", Intl Wkshp Combinatorial Testing, 2016.
3. D.M. Cohen, S.R. Dalal, M.L. Fredman, G.C. Patton, "The AETG System: An Approach to Testing Based on Combinatorial Design", IEEE Trans. Software Eng., vol. 23, no. 7, pp. 437-444, July 1997.
4. C. Yilmaz, S. Fouch´e, M. B. Cohen, A. A. Porter, G. Demiröz, and U. Koc. Moving forward with combinatorial interaction testing. IEEE Computer, 47(2):37–45, 2014.
5. Lei, Y., Kacker, R., Kuhn, D. R., Okun, V., & Lawrence, J. (2007). IPOG: A General Strategy for T-Way Software Testing. 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS07).
6. Pairwise Testing Available Tools. Retrieved February, 2018, from http://www.pairwise.org/tools.asp.

EEG Sinyallerinin Analizi ve Sınıflandırılmasında Yazılım Sistemlerinin Kullanımına Genel Bir Bakış

Buse Nur Uygun[1][0000-0003-1758-5765], Emine Sezer[2], Murat Osman Ünalır[3]

[1] İzmir Yüksek Teknoloji Enstitüsü, Mühendislik Fakültesi, Bilgisayar Mühendisliği, Urla, İzmir
buseuygun@iyte.edu.tr
[2] Ege Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği, Bornova, İzmir
emine.sezer@ege.edu.tr
[3] Ege Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği, Bornova, İzmir
murat.osman.unalir@ege.edu.tr

Özet. Electroencephalogram (EEG), beyinde 2 farklı bölge arasındaki voltaj farkının zaman içerisindeki değişiminin görsel olarak ifade edilmesini sağlayan bir sinyal çeşididir. EEG sinyallerinin nörolojik hastalıkların teşhisinde ve beyinde gerçekleşen psikofizyolojik süreçlerin aydınlatılmasında önemli bir yeri vardır. Son yıllara kadar bu sinyallerin yorumlanması tamamen insan kaynaklı yapılmakta, bu da zaman kaybı ve insani hataya açık olması gibi sorunlara neden olabilmekteydi. Son yıllarda geliştirilen yazılımlar bu tip sorunları en aza indirmektedir. Aynı zamanda yazılım uygulamaları ve sistemler hayata dair tüm alanlarda hızla yer almaktadır. Bu alanlardan biri de hastalık teşhisini de kapsayan sağlık alanıdır. Bu çalışmanın amacı, beyin sinyallerini toplamaya yarayan teknikler hakkında genel bir bilgi vermek, EEG sinyalinin işlenmesinin aşamalarını incelemek ve bu aşamalarda kullanılan farklı tekniklerin kıyaslamasını yapmaktır.

Anahtar Kelimeler: EEG, Dalgacık Dönüşümü, Sağlık Yazılımı, Sinyal İşleme, Hastalık Teşhisi.

A General Approach to Usage of Software Systems in EEG Signal Analysis and Classification

Buse Nur Uygun[1][0000-0003-1758-5765], Emine Sezer[2], Murat Osman Ünalır[3]

[1] Izmir Institute of Technology, Engineering Faculty, Computer Engineering, Urla, İzmir
buseuygun@iyte.edu.tr
[2] Ege University, Engineering Faculty, Computer Engineering, Bornova, İzmir
emine.sezer@iyte.edu.tr
[3] Ege University, Engineering Faculty, Computer Engineering, Bornova, İzmir
murat.osman.unalir@ege.edu.tr

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Abstract. EEG is a signal type that visualize the electrical changes in the two different locations of the brain produced. EEG signals have an important role in diagnosis of neurological diseases and understanding psychophysiologic processes occurring within the brain. Until recent years, interpretation of this signals was completely human driven, this process was time consuming and error prone. recently developed softwares are reducing this kind of problems. At the same time software applications and systems become a big part of number of areas. And one of those area is medicine which contains medical diagnosis. The aim of this study is to give an outline of brain signal acquisition technics, analyze the signal processing stages and compare various techniques that is used during signal processing stages.

Keywords: EEG, Wavelet Transform, Medical Software, Signal Processing, Diagnosis.

1 Giriş

Beyin, yapısal ve kimyasal olarak vücudumuzdaki en karmaşık organdır. Nöronlar, beynin ve sinir sisteminin işlevsel kısmını oluşturan temel hücreleridir. Nöronlar arasında sürekli devam eden bir haberleşme sistemi vardır. Bu haberleşme sisteminin sağlıklı çalışmaması sonucunda akut ve kronik hastalıklar baş gösterir.

Modern görüntüleme teknikleriyle elde edilen beyin sinyallerinin işlenip anlamlı bir veri ortaya çıkarılmasıyla beyinden kaynaklanan bazı hastalıkların teşhisi yapılabilir. Örneğin CT ve MR sayesinde beynin yapısal formu gözlemlenir ya da PET ile farklı zihinsel etkinliklerde beynin hangi bölgesinin kullanıldığı araştırılır, EEG ile epilepsi, depresyon, beyin dolaşım bozuklukları gibi rahatsızlıklar teşhis edilir.

EEG, beyindeki elektriksel aktivitenin elektroensefalogram cihazı kullanılarak analog sinyal olarak kaydedilmesini sağlayan bir tekniktir. Elde edilen sinyal mikrovolt ile ölçülür ve bir yükselticiden geçirilir. Sonrasında elde edilen analog sinyal, bilgisayar tarafından kullanılabilmesi için dijital sinyale çevrilir.

EEG sinyalinin analizi ile kişinin o anki beyinsel faaliyetleri hakkında yararlı bilgiler elde edilir. Ancak sinyalin sadece gözlemleme yoluyla bu bilginin elde edilmesi deneyim gerektirir. Çünkü bu sinyaller yapısı gereği durağan ve doğrusal olmayan sinyallerdir. İçerisinde barındırdığı anlamlı verinin herkes tarafından anlaşılabilmesi için öncelikle gelişmiş sinyal işleme teknikleri uygulanmalıdır. Sonrasında işlenmiş sinyal üzerinden anlamlı özellikler çıkartılabilir [1].

EEG sinyallerinin sadece görsel olarak incelenerek kullanılamamasının başka sebepleri de vardır. EEG görüntüleme sistemleri anlık olarak çok sayıda veri ürettiğinden bunların doktorlar tarafından sürekli bir şekilde gözlemlenip yorumlanması mümkün değildir. Ayrıca sinyalde kolayca göze çarpmayacak değişimler önemli bir hastalığın belirtisi olabilir [2]. Ya da sinyal kaydı alınırken diğer gürültülü sinyaller asıl sinyalin bozulmasına sebep olabilir. Bu durumda istenmeyen gürültülerin asıl sinyalden ayrılması gerekir [3]. Bu gibi problemleri çözebilmek için bilgisayarlı otomasyon sistemlerine ihtiyaç duyulmuştur.

EEG sinyali kullanılarak sınıflandırma yapılabilmesi için EEG sinyalinin sırayla beş aşamadan geçmesi gereklidir. Bu aşamalar sinyalin elde edilmesi, elde edilen

sinyalin gürültülerden temizlenmesi, sinyalin özelliklerinin(özniteliklerinin) çıkartılması, özellik uzayının boyutlarının küçültülmesi ve ardından sınıflandırılmasıdır [4].

Sinyalin elde edilmesinden sonra filtreleme aşaması gelir. Bu aşamanın amacı asıl sinyalin, diğer sinyaller sebebiyle uğradığı bozulmayı en aza indirebilmektir [5].

Sonraki aşama olan öznitelik vektörünün çıkartılması, elde edilen EEG sinyalinin kendine has özelliklerinin ayrıştırılmasını sağlar. Bu sebeple sinyal analizine başlamadan önce sinyalin hangi özniteliklerine ihtiyaç duyulduğuna göre bir sinyal analiz yöntemi seçilmelidir. Sinyal üzerinde uygulanacak analiz yöntemi sinyalin; zaman, frekans, zaman/frekans ve lineer olmayan eksenleri içerisinden hangisi üzerinde incelenmesi gerektiğine göre belirlenmelidir. Örneğin sinyalin frekans ve zaman/frekans ekseninde incelenmesi için öznitelik vektörü çıkarılmadan önce sinyale özel bir dönüşüm uygulanması gerekir. Sinyalin lineer olmayan bir analizle incelenmesi gerekiyorsa higher order spectra, entropi gibi yaklaşımlar kullanılabilir.

Bu analizlerin sonucunda sıra boyut azaltma aşamasına gelir. Boyut azaltma aşamasının amacı, gereksiz öznitelikleri eleyerek sınıflandırma başarısını ve bilgisayar performansını arttırmaktır [6]. Boyut azaltma aşamasından çıkan boyutları indirgenmiş öznitelik vektörü sınıflandırma algoritmasına girdi olarak verilir.

Bu çalışma şu şekilde organize edilmiştir: Giriş bölümünde EEG sinyaline dair genel bilgiler verilmiş, sinyalin işlenmesi sırasında geçtiği aşamalar belirtilmiştir. 2. bölümde çeşitli sinyal elde etme yöntemlerinin karşılaştırılması gösterilmiştir. 3. Bölümde sinyalin filtrelenme aşamasından bahsedilmiştir. 4. bölümde EEG sinyalinin özniteliklerinin çıkartılması detaylı bir şekilde anlatılmıştır. 5. bölümde boyut azaltma aşamasında kullanılan yaygın algoritmalar ve bu alanda yapılan bazı çalışmalar açıklanmıştır. 6. Bölümde sınıflandırma aşamasında yaygın olarak kullanılan algoritmalar açıklanmıştır ve bu algoritmaların sinyal sınıflandırma amacıyla kullanımları kıyaslanmıştır. 7. Bölüm ise sonuç kısmını içermektedir.

2 Sinyal Elde Etme Yöntemleri

Beyin sinyalinin elde edilmesi için çeşitli yöntemler kullanılır. Bu aşamada kullanılan yöntemler genel olarak invasiv ve invasiv olmayan yöntemler olarak ikiye ayrılır. Bu ayrım hastaya yapılacak müdahalenin derinliğini belirtmek için kullanılır. Beyin sinyali elde edilirken başvurulan invasiv yöntemler intracranial electroencephalography (EcoG) ve intra-cortical electrode array'dir. İnvasiv olmayan yöntemler ise electroencephalography (EEG), functional magnetic resoncance imaging (fMRI), near infrared spectroscopy (NIRS), positron emission tomography (PET), magnetoencephalography (MEG) olarak sayılabilir [7]. Her sinyal ölçme metodu farklı amaca yönelik kullanılır ve çeşitli avantaj ve dezavantaja sahiptir [8]. Tablo 1 üzerinde invasiv olmayan metotların karşılaştırılması gösterilmiştir.

3 Sinyalin Filtrelenmesi

Kaydedilen EEG sinyallerinin içerisinde çeşitli kaynaklardan gelen gürültüler bulunur. Bu sebeple incelenecek asıl sinyalin amacına uygun bir şekilde, asıl sinyalden

Tablo 1. EEG, fMRI, NIRS, MEG metotlarının karşılaştırılması

Metot	Kaydedilen sinyal türü	Avantajları	Dezavantajları
EEG	Kafatası üzerinden kaydedilen beyin sinyalleri kullanılır.	Yüksek zamansal çözünürlüğe sahiptir. Güvenli ve kolay uygulanabilen bir tekniktir.	Gürültü yüzünden sinyalin temizlenmesi gereklidir. Boyutsal olarak çözünürlüğü düşüktür.
fMRI	BOLD dönütünü kullanarak oluşturulan metabolik sinyaller kullanılır.	Yüksek zamansal ve boyutsal çözünürlüğe sahiptir.	Kurulum maliyeti fazladır. Veri alımında gecikmeler yaşanır.
NIRS	BOLD dönütünü kullanarak oluşturulan metabolik sinyaller kullanılır.	Yüksek boyutsal çözünürlüğe sahiptir. Maliyeti düşüktür. Taşınabilir.	Düşük zamansal çözünürlüğe sahiptir. Performansı daha düşüktür.
MEG	Elektriksel aktivite sonucu oluşturulan manyetik sinyaller kullanılır.	Daha geniş frekans aralığına sahiptir. Zamansal ve boyutsal çözünürlüğü oldukça iyidir.	Kullanılan cihaz ağır ve hantaldır. Deneylerin maliyeti yüksektir.

istenmeyen gürültüler ayrıştırılmalıdır. Çeşitli gürültü tiplerine örnek verilecek olursa: Göz hareketlerine bağlı olarak gerçekleşen gürültü sinyali [9], kas hareketlerine bağlı olarak gerçekleşen gürültü sinyali [10], kalp kası hareketine bağlı olarak gerçekleşen gürültü sinyali [11], çevredeki makinelerin neden olduğu sinyaller olarak sıralanabilir.

Filtreleme işleminin incelenmek istenen sinyalin öznitelikleri bozulmadan yapılması için çeşitli yöntemler vardır. Bu yöntemlerden biri frekans seçici filtreler kullanılmasıdır. Frekans seçici filtreler; yüksek geçiren, alçak geçiren, bant geçiren filtreler olarak ayrılırlar. Çalışma mantıkları elde edilmek istenen sinyalle gürültü sinyalinin frekans bantlarının farklı olduğu varsayımına dayanır. Bu sebeple frekans seçici filtreler geniş frekans bandındaki gürültüleri temizlemekte başarılı değildir [9].

Gürültü temizlenmesinde başka bir yaklaşım da temizlenecek olan sinyalin analiz edilmesidir. Bu sayede elimizdeki gürültü sinyalinin frekansının bilgisiyle, asıl sinyal ile gürültü sinyali birbirinden ayrıştırılabilir. Bunun sağlanabilmesi için sinyal; zaman, frekans eksenlerinde incelenebilir ya da adaptif filtreleme, Wiener filtreleme yöntemi, bağımsız bileşen analizi algoritması kullanılabilir [10].

4 Sinyalin Özniteliklerinin Çıkartılması

Sinyalin elde edilmesinden sonra sinyalin özniteliklerinin çıkartılması aşaması gelir. EEG sinyallerinde farklı frekans bandına sahip beş ayrı dalga bulunur. Bu dalgalar: delta (1–4 Hz), teta (4–8Hz), alfa (8–13 Hz), beta (13–30Hz) ve gama (30 Hz ve üstü) dalgaları olmak üzere beşe ayrılır[11]. Sinyal analizi aşamasında farklı frekans alt bantlarına denk gelen bu beyin dalgalarından yararlanılır. Beyin dalgaları kişinin mental durumuna, yaşına, uyku ve uyanıklık durumuna ya da sağlık durumuna göre değişkenlik gösterir [11–13].

Sinyal analizinde ilk zamanlar Fourier dönüşümü (FD) kullanılmıştır. FD yapısı gereği, EEG sinyalini zaman ekseninde ele almadığı için sinyalin durağan olduğunu varsayar. Ancak EEG'nin lineer olmayan ve dinamik bir yapısı vardır. Bu sebeple EEG'ye daha uygun olarak zaman eksenini de analiz içerisine dahil eden başka dönüşüm metotları kullanılmalıdır [14]. Bu dönüşüm metodu çeşitlerine kısa zamanlı Fourier dönüşümü (KZFD) ve dalgacık dönüşümü (DD) örnek verilebilir.

4.1 Kısa Zamanlı Fourier Dönüşümü

FD, zaman ekseninin hesaba katılmasının önemli olduğu sinyal çeşitlerinin analizinde kullanılmaya uygun değildir. Bu sebeple KZFD metodu geliştirilmiştir. KZFD'de durağan olmayan sinyal, birbiri ardına geldiği kabul edilen bölütlere ayrılır ve bu bölütlerin durağan olduğu varsayılarak her bir bölüte FD uygulanır. Bölütler oluşturulurken boyutu, oluşturulan bölütler kadar olan bir pencereleme fonksiyonundan yararlanılır. Ancak pencere fonksiyonunun boyutunun seçiminde zaman ya da frekans çözünürlüğünün birinden feragat edilmesi gerekmektedir [15].

4.2 Dalgacık Dönüşümü

Dalgacık dönüşümünde de bir pencereleme fonksiyonu kullanılır. Ana dalgacık denen bu pencereleme fonksiyonunun üstünlüğü ise ölçeklenebilir olmasından gelir. Bu sayede dalgacık dönüşümü, KZFD'de görülen pencere boyutu seçimi yüzünden meydana gelen hassasiyet kaybı sorununun çözümüne olanak sağlar ve sinyalin zaman/frekans ekseninde incelenmesini mümkün kılar. Böylelikle durağan olmayan sinyallerin işlenmesi mümkün olur[17]. Dalgacık dönüşümünün iki türü vardır. Bunlar ayrık dalgacık dönüşümü (ADD) ve sürekli dalgacık dönüşümüdür (SDD) [18].

Sürekli Dalgacık Dönüşümü. SDD, zaman içerisinde ana dalganın ölçeklenmesi ve kaydırılmasıyla değişen değerinin sinyalle çarpılması ile elde edilen sonuçlarının toplanması ile bulunur. Bu işlemlerin sonucunda dalgacık katsayıları elde edilir. Ancak SDD'de ölçeklendirme ve kaydırma parametreleri sürekli değiştiğinden her değişen parametre değeri için dalgacık katsayılarının tekrardan hesaplanması zaman alan bir işlemdir.

Ayrık Dalgacık Dönüşümü. ADD'de ölçeklendirme ve kaydırma parametreleri ikinin üstleri olacak şekilde seçilir. Bu şekilde sinyal, alt bantlarına ayrılırken çıkan sonuçlar SDD ile aynı olmakla birlikte sonuçlar daha hızlı alınmaktadır.

Sinyal, dalgacık dönüşümü ile alt frekans bantlarına ayrışır. Bunu sağlamak için de sinyale filtreler uygulanır. Yüksek frekans bileşenlerin analizi yüksek geçiren filtreden geçirilerek yapılır. Böylece düşük frekans bileşeni elenmiş olur ve ayrıntı bileşeni elde edilir. Düşük frekans bileşenlerin analizi ise tam tersi olarak, düşük geçiren filtreden geçirilerek yapılır. Böylece yüksek frekans bileşeni elenir ve sonucunda yaklaşıklık bileşeni elde edilir. Dalgacık dönüşümünde sürekli değişen frekans alt bantları

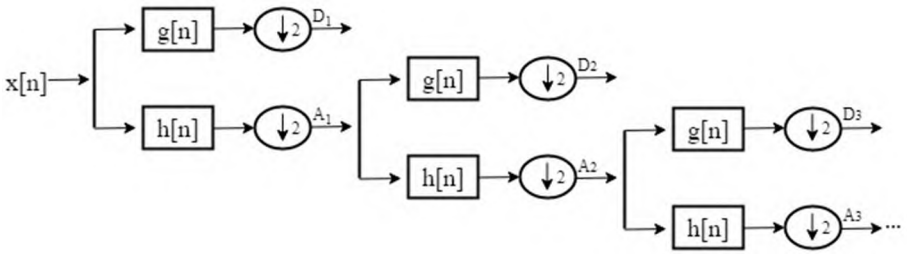
için yüksek ve alçak geçiren filtreler çalıştığı için sinyal yaklaşıklık ve ayrıntı bileşenlerine ayrılabilir. Bu işleme çoklu çözünürlük ayrışımı ismi verilir. Seçilen ayrışma seviyesi kadar bu işlem devam eder. Şekil 1'de üç ayrışma seviyesi kadar alt bantlarına ayrılmış örnek bir frekans(x[n]) gösterilmektedir.

Analizden sonra yapılacak sınıflandırmaya uygun olacak şekilde ana dalgacık belirlenirken genel amaçlı kullanılabilen bir dalga seçilebilir. Örneğin epileptik EEG sinyalinin sınıflandırılmasında Daubechies dalgası iyi bir başarı yüzdesi verdiği için genel amaçlı olarak kullanılabilen bu dalga türünün seçilmesi uygundur [19].

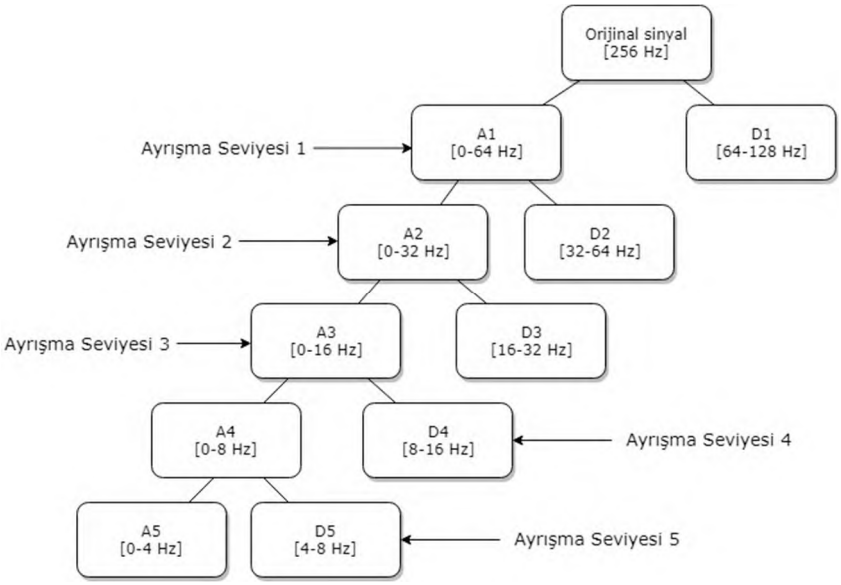
Ayrışma seviyesi seçimi de ayrı bir problem olarak ele alınmalıdır. Temelde ayrışma seviyesi arttıkça sinyale dair daha ayrıntılı bilgiler elde ederiz ancak bilgisayara getirdiği ekstra yükü de arttırmış oluruz. Ayrışma seviyesi belli bir değerin üzerine çıktığında başarı yüzdesi nerdeyse hiç artmadığı gibi, bilgisayar performansı da bir hayli düşmektedir [20]. Ayrıca ayrışma seviyesinin çok arttırılması sınıflandırma yapılırken kullanılacak öznitelik vektörünün boyutunu arttırdığı için sınıflandırma başarısının düşmesine sebep olur. Optimal değerin bulunabilmesi için incelenen sinyalin frekans özellikleri dikkate alınmalıdır [21]. EEG sinyalinde 30 Hz'in üzerinde yararlı bir frekans bileşeni olmadığı için 256 Hz'lik bir sinyalde ayrışma seviyesi beş olarak belirlenebilir [22]. Ayrışma seviyesi 5'in altında seçilirse düşük frekans bandında hassasiyet kaybı olacaktır. Örneğin delta dalgasının temsili zorlaşacaktır. Ayrışma seviyesinin 5'in üzerinde seçilmesi ise gereksizdir çünkü tüm EEG dalgalarının temsili bu seviyede mümkündür [23].

Alt bantlarına ayırma işlemi sonucu oluşan A5 bileşeni delta aralığını (1-4 Hz), D5 bileşeni teta aralığını (4-8 Hz), D4 bileşeni alfa aralığını (8-13 Hz) ve D3 bileşeni beta aralığını (13-30) yansıtmaktadır. D1 ve D2 bantları 30Hz'in üzerinde bir frekans aralığına sahip olduğu için, diğer bantlara kıyasla daha az bilgi içermektedirler. Bu sebeple yapılan benzer çalışmalarda D3, D4, D5 ayrıntı alt bantları ve A5 yaklaşım alt bantları kullanılarak inceleme yapılmaktadır [24]. Şekil 2'de 256 Hertzlik bir frekansın alt bantlarına ayrışması gösterilmektedir.

Dalgacık dönüşümü ile frekans alt bantlarına ayrılmış EEG sinyalinden öznitelik vektörleri elde edilir. Öznitelik vektörünün oluşturulabilmesi için frekans alt bantları üzerinden dalgacık katsayılarının dağılımını gösteren istatistiki değerler hesaplanır [25]. Bu sayede EEG'nin zaman frekans dağılımı gösterilir. Bu işlemler sırasında kullanılan bazı istatistik değerler şunlardır: Her bir alt banttaki katsayıların mutlak değerlerinin ortalaması, katsayıların mutlak değerlerinin maksimumu, katsayıların kuvvetlerinin ortalaması, katsayıların standart sapması.



Şekil 1. Dalgacık dönüşümü ile sinyalin alt bantlarına ayrılması



Şekil 2. 256 Hz'lik bir sinyalin alt bantlarına ayrışması

5 Öznitelik Vektörünün Boyutlarının Azaltılması

Öznitelik vektöründeki her özellik sınıflandırma algoritmasına verilmez. Bunun sebebi sinyalin içerisinde sınıflandırma başarısını düşürecek gürültülü veri ya da ilgisiz özelliklerin bulunmasıdır. Bu sebeple öznitelik vektörünün çıkartılma aşamasından sonra boyut azaltma aşaması gelir. Bu aşamada sinyali en iyi temsil edecek özelliklerin çıkartılabilmesi için boyut azaltma ve özellik seçme algoritmaları kullanılır. Teorik olarak, çıkarılan tüm özelliklerin kombinasyonları ile sınıflandırma yapılarak en yüksek başarıya sahip kombinasyondaki öznitelikler seçilebilir. Ancak öznitelik vektörü genişse bu deneme süresi çok uzun sürer. Sürenin kısaltılması için genetik algoritmalar, temel bileşen analizi gibi yöntemler kullanılabilir [26].

Bu aşamada en uygun özniteliklerin seçilmesi için kullanılan yaygın lineer metotlara örnek olarak temel bileşenler analizinin (TBA) yanı sıra bağımsız bileşenler analizi (BBA) ve doğrusal ayrım analizi (DAA) verilebilir [27-28]. Bu üç algoritmanın destek kuvvet makinesi sınıflandırıcısı ile beraber kullanıldığı çalışmada en yüksek başarı oranının sırasıyla DAA, BBA, TBA olduğu gösterilmiştir [22].

5.1 Temel Bileşenler Analizi

Temel bileşenler analizi, özellik çıkarımı ve boyut azaltılması için yaygın olarak kullanılan bir istatiksel metottur. Bu metot ile, d boyutlu bir veri daha az bir boyutta ifade edilmeye çalışılır. Bu sayede zaman ve yer karmaşıklığının azaltılması hedeflenir.

Dönüşüm sonrasında elde edilen değişkenler ilk değişkenlerin temel bileşenleri olarak adlandırılır. İlk temel bileşen olarak varyans değeri en büyük olan seçilir ve diğer temel bileşenler varyans değerleri azalacak şekilde sıralanır. Bu teknik, özellikle çoklu kaynaklardan gelen sinyallerin bölütlenmesinde kullanılır. TBA çeşitli çalışmalarda EEG sinyalinden çıkartılan öznitelik vektörünün boyutlarının küçültülmesinde kullanılmıştır [29].

5.2 Bağımsız Bileşenler Analizi

Bağımsız bileşenler analizi, birbirinden bağımsız çok değişkenli sinyalleri, bağımsız bileşenlerin doğrusal birleşimi şeklinde ifade etmeye çalışır. X ve y iki değişken olmak üzere herhangi birinin değeri diğerinin değerine bağlı değilse bunlara bağımsız değişken denir. Bu analizde amaç, bağımsız 2 kaynaktan gelen sinyallerin birbirinden ayrıştırılmasıdır. BBA çeşitli çalışmalarda, EEG sinyalinden çıkartılan öznitelik vektörünün boyutlarının küçültülmesinde ya da sinyalin sınıflandırılmasında kullanılmıştır [29-30].

5.3 Doğrusal Ayrım Analizi

Doğrusal ayrım analizinde amaç, sınıfları en iyi şekilde arayan lineer doğruyu bulmaktır. Algoritma bunun için ilk başta belirlenen grupların arasındaki mesafeyi maksimum yapmaya çalışır [6].

6 Sınıflandırma Aşaması

Son olarak, boyutu azaltılmış sinyalin sınıflandırılması aşaması gelir. Sinyal sınıflandırma aşamasında çeşitli makine öğrenmesi algoritmaları kullanılabilir. Bu algoritmalar doğrusal sınıflandırıcılar, sinir ağları, doğrusal olmayan Bayesian sınıflandırıcılar, en yakın komşu sınıflandırıcıları ve bu algoritmaların kombinasyonları olarak 5'e ayrılarak incelenebilir. [31].

6.1 Doğrusal Sınıflandırıcılar

Doğrusal sınıflandırma algoritmaları, veri setindeki elemanları sınıflara ayırabilmek için doğrusal fonksiyonlardan yararlanır. En yaygın kullanılan 2 çeşidi doğrusal ayrım analizi (5.3) ve destek kuvvet makinesi (DKM) algoritmalarıdır.

Destek Kuvvet Makinesi Algoritması. Destek kuvvet makinesi, sınıfları birbirinden ayırabilmek için doğrusal bir fonksiyondan yararlanır. DKM'nin diğer doğrusal sınıflandırıcı algoritmalarından farkı, marjin değerini (ayrılma sağlayan fonksiyonun her iki tarafındaki en yakın komşuların bu fonksiyona olan uzaklığı) en uzak yapan destek vektörlerini bulmaya çalışmasıdır. DKM, doğrusal ayrılan ve doğrusal ayrılmayan veriler için kullanılabilir. Doğrusal ayrılan veriler için uygulaması yukarıda anlatıldığı

gibidir, doğrusal ayrılmayan veriler için önce veriler doğrusal ayrılabilecekleri bir uzaya aktarılır, sonrasında sınıflandırma işlemi yapılır. DKM, sınıflandırma sırasında teorik riski sıfıra indirmeye çalıştığı için başarı yüzdesi yüksektir. Ayrıca eğitim süresinin uzun olduğu durumlarda ve gelen özellik vektörünün boyutunun fazla olması durumlarında bile iyi sonuç verir [32-33].

6.2 Yapay Sinir Ağları

Doğrusal sınıflandırma algoritmaları ile birlikte beyin-bilgisayar arayüzü araştırmalarında en sık kullanılan sınıflandırma algoritmalarındandır. Sinir ağlarının çalışma prensibi, beyindeki nöronların yapay olarak modellenerek lineer olmayan sınıflandırma için kullanılmasıdır. Sinir ağlarının en sık kullanılan çeşidi Çok Katmanlı Perceptron (ÇKP)'dir. ÇKP dışında başka sinir ağları mimarileri de mevcuttur.

Çok Katmanlı Perceptron. Çok katmanlı perceptron, yapay sinir ağlarının yapıtaşı olan nöronların birden çok katman halinde bulunmasından meydana gelir. Giriş ve çıkış katmanlarının arasındaki katmanlara gizli katman denir. Sadece komşu katmanlar birbirleriyle iletişim halindedir. Çıkış katmanı, girdinin sınıfının ne olacağını belirtir. Ancak ÇKP'nin eğitim süresi uzun tutulduğunda gelen farklı verileri doğru sınıflandırmada başarısız olduğu görülmüştür [34]. Bu sebeple ÇKP'nin tasarımı yapılırken girilen parametre değerleri dikkatli seçilmelidir.

Diğer Sinir Ağları Mimarileri. Sinyal sınıflandırma amacıyla çeşitli sinir ağı mimarileri kullanılmaktadır. Bunlara örnek olarak Gaussian sınıflandırma algoritması, learning vector quantization, bulanık ARTMAP, dinamik sinir ağı, radyal tabanlı sinir ağları sayılabilir.

6.3 Doğrusal Olmayan Bayesian Sınıflandırma Algoritmaları

Bu sınıflandırma algoritmalarına saklı Markov model (SMM) ve Bayes quadratic algoritmaları verilebilir. Bu sınıflandırma yöntemleri ile veriler doğrusal olmayan bir şekilde sınıflandırılabilir.

Bayes Quadratic. Bayesian sınıflandırmasında, sisteme sunulan öğretilmiş veriler kullanılarak yapılan olasılık işlemleri ile yeni gelen veriye atanacak sınıf belirlenir. Ancak sisteme sunulan eğitim verisinin bir kategorisi olmalıdır.

Saklı Markov Model. SMM, yaygın olarak zaman serilerinin sınıflandırılmasında kullanılan bir algoritmadır ve özellikle ses tanıma alanında sıkça kullanılır [35]. Markov modeliyle her stokastik olayın olasılık değeri modellenebilir. Böylece t anındaki durum bilgisi ile t+1 anındaki durum tahminlenir. SMM'nin EEG sinyali ile yapılan uyku safhası sınıflandırılması için kullanıldığı çalışmalarda başarılı sonuçlar elde edilmiştir [36-37].

6.4 En Yakın Komşu Sınıflandırması

Sınıflandırılacak olan yeni elemanın, ona en yakın elemanların sınıfı göz önüne alınarak bir sınıfa dahil edilmesi mantığıyla çalışır. Veri setinin içinden en yakın elemanların seçimi metrik bir değere göre belirleniyorsa bu sınıflandırma algoritmasına k en yakın komşu algoritması denir. Bu sınıflandırma metodunun dezavantajları tembel bir öğrenme algoritması olması ve k değerinin seçiminde bir belirsizlik olmasıdır [38]. Ayrıca geniş veri setlerinde çalışma performansı düşüktür. Bu sebeplerden dolayı sinyal sınıflandırma alanında sık kullanılmamaktadır.

7 Sonuç

EEG, beynin anlık durumunu ve beyinle ilgili olan hastalıkların teşhis edilebilmesi için gerekli olan verilerin kolay bir şekilde elde edilmesini sağlar. Çeşitli mental durumların (yaş, cinsiyet, uyku/uyanıklık durumu) EEG üzerinde doğrudan etkisi vardır. EEG sinyallerinden elde edilen bilgilerle patolojik sinyallerin sınıflandırması yapılabilir. Bunun için EEG sinyallerinin ilk olarak işlenmesi gerekir. Sinyal işlenmesi sürecinde çeşitli dönüşümler kullanılır. Ancak EEG sinyallerinin doğasına uygun olacak şekilde işlenebilmesi için kullanılacak yöntemlerde seçici davranılmalıdır. Sinyal analizinde sık kullanılan bir yöntem dalgacık dönüşümüdür. Bu dönüşüm, EEG sinyaline geniş bir bakış açısı sağlar. Dalgacık dönüşümü sonrasında uygulanan istatiksel hesaplamalarla elde edilen öznitelik vektörü sinyalin kendine has özelliklerini belirtir. Yapılacak sınıflandırmanın başarısı arttırmak amacıyla, oluşturulan vektörün boyutunu indirgeyen doğrusal olan ve doğrusal olmayan algoritmalar kullanılır. Boyutu indirgenmiş öznitelik vektörü, sınıflandırılması için bir makine öğrenmesi algoritmasına girdi olarak verilir.

Bu çalışmada bilgisayarlı otomasyon sistemlerinin EEG sinyalinden anlamlı bilgi elde edilmesi amacıyla kullanılması ve bu süreçte EEG sinyalinin geçtiği aşamalar üzerinde durulmuştur. Sinyallerden kendine has özniteliklerinin elde edilebilmesi için uygun analiz metotları kullanılmalıdır. Bu sebeple bu çalışmada EEG sinyalinin analizi zaman/frekans ekseninde yapılmıştır.

Kaynakça

1 D. P. Subha, P. K. Joseph, R. Acharya U, and C. M. Lim, “EEG signal analysis: a survey.,” *J. Med. Syst.*, vol. 34, no. 2, pp. 195–212, 2010.

2 D. M. Murray, G. B. Boylan, I. Ali, C. A. Ryan, B. P. Murphy, and S. Connolly, “Defining the gap between electrographic seizure burden, clinical expression and staff recognition of neonatal seizures,” *Arch. Dis. Child. Fetal Neonatal Ed.*, vol. 93, no. 3, pp. 187–191, 2008.

3 R. Romo Vázquez, H. Vélez-Pérez, R. Ranta, V. Louis Dorr, D. Maquin, and L. Maillard, “Blind source separation, wavelet denoising and discriminant analysis for EEG artefacts and noise cancelling,” *Biomed. Signal Process. Control*, vol. 7, no. 4,

pp. 389–400, 2012.

4 T. Lajnef *et al.*, “Learning machines and sleeping brains: Automatic sleep stage classification using decision-tree multi-class support vector machines,” *J. Neurosci. Methods*, vol. 250, pp. 94–105, 2015.

5 M. Ronzhina, O. Janoušek, J. Kolářová, M. Nováková, P. Honzík, and I. Provazník, “Sleep scoring using artificial neural networks,” *Sleep Med. Rev.*, vol. 16, no. 3, pp. 251–263, 2012.

6 S. F. Liang, H. C. Wang, and W. L. Chang, “Combination of EEG complexity and spectral analysis for epilepsy diagnosis and seizure detection,” *EURASIP J. Adv. Signal Process.*, vol. 2010, pp. 17–20, 2010.

7 S. V. Bandara, “Brain signal acquisition methods in BCIs to estimate human motion intention – a survey Brain signal acquisition methods in BCIs to estimate human motion intention – a survey,” no. December, 2018.

8 M. R. Lakshmi, T. V. Prasad, and V. C. Prakash, “Survey on EEG signal processing methods,” *Sci. Comput. Eng. Softw.*, vol. 4, no. 1, pp. 84–91, 2014.

9 A. van Boxtel, P. Goudswaard, and L. R. B. Schomaker, “Amplitude and Bandwidth of the Frontalis Surface EMG: Effects of Electrode Parameters,” *Psychophysiology*, vol. 21, no. 6, pp. 699–707, 1984.

10 S. Motamedi-Fakhr, M. Moshrefi-Torbati, M. Hill, C. M. Hill, and P. R. White, “Signal processing techniques applied to human sleep EEG signals - A review,” *Biomed. Signal Process. Control*, vol. 10, no. 1, pp. 21–33, 2014.

11 R. Acharya U., O. Faust, N. Kannathal, T. Chua, and S. Laxminarayan, “Non-linear analysis of EEG signals at various sleep stages,” *Comput. Methods Programs Biomed.*, vol. 80, no. 1, pp. 37–45, 2005.

12 M. M. M. Rizon and R. N. S. Yaacob, “EEG Feature Extraction for Classifying Emotions using FCM and FKM 2 Materials and Methods,” *Ratio*, no. June, pp. 299–304, 2008.

13 H. P. Landolt and A. A. Borbély, “Age-dependent changes in sleep EEG topography,” *Clin. Neurophysiol.*, vol. 112, no. 2, pp. 369–377, 2001.

14 A. T. Tzallas, M. G. Tsipouras, and D. I. Fotiadis, “Epileptic Seizure Detection in EEGs Using Time–Frequency Analysis,” *{IEEE} Trans. Inf. Technol. Biomed.*, vol. 13, no. 5, pp. 703–710, 2009.

15 L. Guo, D. Rivero, J. Dorado, C. R. Munteanu, and A. Pazos, “Automatic feature extraction using genetic programming: An application to epileptic EEG classification,” *Expert Syst. Appl.*, vol. 38, no. 8, pp. 10425–10436, 2011.

16 H. Adeli, Z. Zhou, and N. Dadmehr, “Analysis of EEG records in an epileptic patient using wavelet transform,” *J. Neurosci. Methods*, vol. 123, no. 1, pp. 69–87, 2003.

17 Y. Kumar, M. L. Dewal, and R. S. Anand, “Epileptic seizure detection using DWT based fuzzy approximate entropy and support vector machine,” *Neurocomputing*, vol. 133, pp. 271–279, 2014.

18 H. Ocak, “Automatic detection of epileptic seizures in EEG using discrete wavelet transform and approximate entropy,” *Expert Syst. Appl.*, vol. 36, no. 2 PART 1, pp. 2027–2036, 2009.

19 A. Subasi, “Automatic detection of epileptic seizure using dynamic fuzzy neural networks,” *Expert Syst. Appl.*, vol. 31, no. 2, pp. 320–328, 2006.

20 T. Gandhi, B. K. Panigrahi, and S. Anand, “A comparative study of wavelet families for EEG signal classification,” *Neurocomputing*, vol. 74, no. 17, pp. 3051–3057, 2011.
21 I. Transactions and I. Technology, “Multiclass Support Vector Machines for EEG-Signals Classification,” vol. 11, no. April 2007, 2016.
22 A. Subasi and M. I. Gursoy, “EEG signal classification using PCA, ICA, LDA and support vector machines,” *Expert Syst. Appl.*, vol. 37, no. 12, pp. 8659–8666, 2010.
23 B. Şen, M. Peker, A. Çavuşoğlu, and F. V Çelebi, “A comparative study on classification of sleep stage based on EEG signals using feature selection and classification algorithms.,” *J. Med. Syst.*, vol. 38, no. 3, p. 18, 2014.
24 X. W. Wang, D. Nie, and B. L. Lu, “Emotional state classification from EEG data using machine learning approach,” *Neurocomputing*, vol. 129, pp. 94–106, 2014.
25 P. Jahankhani, V. Kodogiannis, and K. Revett, “EEG signal classification using wavelet feature extraction and neural networks,” *Proc. - IEEE John Vincent Atanasoff 2006 Int. Symp. Mod. Comput. JVA 2006*, pp. 120–124, 2006.
26 H. Ocak, “Optimal classification of epileptic seizures in EEG using wavelet analysis and genetic algorithm,” *Signal Processing*, vol. 88, no. 7, pp. 1858–1867, 2008.
27 L. J. Cao, K. S. Chua, W. K. Chong, H. P. Lee, and Q. M. Gu, “A comparison of PCA, KPCA and ICA for dimensionality reduction in support vector machine,” *Neurocomputing*, vol. 55, no. 1–2, pp. 321–336, 2003.
28 X. Wang and K. K. Paliwal, “Feature extraction and dimensionality reduction algorithms and their applications in vowel recognition,” *Pattern Recognit.*, vol. 36, no. 10, pp. 2429–2439, 2003.
29 L. Zhukov, D. Weinstein, and C. Johnson, “Independent Component Analysis for EEG Source Localization In Realistic Head Models,” *IEEE Eng. Med. Biol. Mag.*, vol. 19, no. 3, pp. 87–96, 2000.
30 C. Papadelis *et al.*, “Monitoring sleepiness with on-board electrophysiological recordings for preventing sleep-deprived traffic accidents,” *Clin. Neurophysiol.*, vol. 118, no. 9, pp. 1906–1922, 2007.
31 F. Lotte, M. Congedo, A. Lécuyer, F. Lamarche, and B. Arnaldi, “A review of classification algorithms for EEG-based brain-computer interfaces,” *J. Neural Eng.*, vol. 4, no. 2, 2007.
32 A.K. Jain, R.P.W. Duin, and J. Mao, “Statistical Pattern Recognition: A Review,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 4–37, 2000.
33 G. D., P. D. A., A. C. W., and T. M. H., “Comparison of linear, nonlinear, and feature selection methods for EEG signal classification,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 11, no. 2, pp. 141–144, 2003.
34 D. Arpit *et al.*, “A Closer Look at Memorization in Deep Networks,” 2017.
35 Rabiner,L.R., “A tutorial on hidden Markov models and selected applications in speech recognition,” *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
36 M. Längkvist, L. Karlsson, and A. Loutfi, “Sleep Stage Classification Using Unsupervised Feature Learning,” *Adv. Artif. Neural Syst.*, vol. 2012, pp. 1–9, 2012.
37 A. Flexer, G. Gruber, and G. Dorffner, “A reliable probabilistic sleep stager based on a single EEG signal,” *Artif. Intell. Med.*, vol. 33, no. 3, pp. 199–207, 2005.
38 G. Yang, M. Kifer, and C. Zhao., “KNN Model-Based Approach in Classification,” *Int. Conf. Ontol. Databases, Appl. Semant.*, no. January, pp. 671–688, 2003.

Laboratuvar Bilgi Sistemlerinin Bulut Dönüşümü

Bengi Tuğcu İdemen ve Arhun Hınçalan

Panates Bilişim Teknoloji Araştırma ve Geliştirme Merkezi
b.idemen@panates.com, a.hincalan@panates.com

Özet. Sağlık hizmetlerinde doktorların hastalara doğru tanı vermesi ve tedaviye doğru devam edebilmesi konusunda ihtiyaç duydukları en önemli bilgi kaynağı, hastalara tanı amacıyla yapılan tetkiklerdir. Bu nedenle laboratuvar süreçleri etkin sağlık hizmeti açısından büyük önem taşımaktadır. Laboratuvarlarda çok sayıda ve farklı türlerde sayısal veri üretilmektedir. Laboratuvar süreçlerinin sonunda elde edilecek olan sağlık verisine; insan, makine (cihaz), kullanılan kit ve malzeme, havadaki nem sıcaklık ışık vb. gibi çok sayıda faktörler etki etmektedir. İnsan gözünün bu kadar etmeni değerlendirerek çıkan sonucun doğruluğunu ve ne oranda doğru olduğunu ölçümleyebilmesi ve teyit edebilmesi mümkün değildir. Bu sorunlara çözüm üretmeyi hedefleyen bu çalışma ile, geleneksel laboratuvar bilgi sistemlerinin buluta taşınarak laboratuvar ortamlarının sensörler aracılığıyla IoT (Nesnelerin İnterneti) adaptasyonu, dış etkenlerin laboratuvar testleri üzerindeki etkilerinin izlenebilirliği ve tetkik sonuçlarının kalitesinin artırılması amaçlanmaktadır.

Anahtar Kelimeler: Laboratuvar Bilgi Sistemi (LIS), Bulut, IoT (Nesnelerin İnterneti)

“Laboratory Information System” Transfer to Cloud

Bengi Tuğcu İdemen and Arhun Hınçalan

Panates Information Technologies Research and Development Center
b.idemen@panates.com, a.hincalan@panates.com

Abstract. In health services, the most important information source that doctors need for accurate diagnosis and continuation of treatment is the tests performed on patients. Therefore, laboratory processes are of great importance for effective health care. Numerous and different types of numerical data are produced in laboratories. In laboratory processes, human, machine (device), used kit and material, air humidity, temperature, light etc. as well as a number of factors affect the final output data. It is not possible for the human eye to evaluate and verify the accuracy of the result and to what extent it is correct by evaluating such factors. At this point, the aim of this study is to increase the quality of the test results and the adaptation of IoT (Internet of Things) to the laboratory envi-

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

ronments by the transfer of traditional laboratory information systems to the cloud and the effects of external factors on laboratory tests.

Keywords: Laboratory Information System (LIS), Cloud, IoT (Internet of Things)

1 Giriş

Teknoloji, günümüzde herkesin hayatını etkilemekte dolayısıyla hem insanlar hem de işletmeler içinn vazgeçilmez bir unsur haline gelmiştir. Maddi faydaların yanı sıra bireylerin beklentilerini yükselterek manevi kazanımlar da sunmaktadır. Teknolojinin faydalarının görüldüğü en önemli alanlardan biri sağlık sektörüdür. Hastaneler ve sağlık kuruluşları, sayısız karmaşık işlem ve süreçleri içeren oldukça büyük ve heterojen sistemlerdir. Teknoloji ve tıp bilgilerinin birleşmesi, bu karmaşıklığı daha da arttırmaktadır. Böyle bir karmaşıklıkla birlikte muazzam miktarda bilgi üretilmektedir. Sağlık Bilgi Sistemleri (Health Information System - HIS), bireylerin sağlığına, klinik bakımına veya sağlıkla ilgili kuruluşların faaliyetlerine ilişkin bu büyük miktarda veriyi toplayan, saklayan, yöneten veya ileten bilgisayar sistemleridir [1].

Sağlık bilişimindeki en önemli parçalarından biri olan Laboratuvar Bilgi Sistemleri (Laboratory Information Systems - LIS), test kararlarından başlayarak raporlama işlemine kadar izleyen ve yöneten bilgi yönetim sistemidir [2]. LIS, laboratuvarların çok çeşitli bilgi işlem ihtiyaçlarını destekleyen karmaşık, birbiriyle ilişkili bilgisayar programları ve alt yapıdan oluşur.

Sağlık kurumlarında kullanılan Laboratuvar Bilgi Sistemleri (LIS), çağın ihtiyaç ve gereksinimlerinden zamanla uzaklaşmış olup günümüz teknolojilerinin oldukça gerisinde kalmış durumdadır. LIS'ler, numune ve test sırası alımı, numune işleme ve izleme, analiz ve yorum desteği ve rapor oluşturma ve dağıtımı dahil olmak üzere hasta testinin tüm aşamalarında fonksiyonlara sahiptir. Ek olarak, LIS'ler laboratuvarların faaliyetlerini yürütmeleri ve sürekli iyileştirme ve kalite girişimlerini desteklemek için ihtiyaç duydukları yönetim raporları ve diğer verileri sağlar [3].

Laboratuvar bilgi sistemleri genel olarak aşağıdaki fonksiyonları desteklemektedir [4]:

- Test isteklerini kaydeder.
- Örnek toplama ve işleme sürecini programlar ve izler.
- Tamamlanan test sonuçlarını kaydeder.
- Hasta kayıtlarının bir parçası olacak test raporlarını üretir.
- Takip edilecek uyaranlar üretir.
- Hastalara uygulanan testler için periyodik özet raporlar oluşturur.
- Hasta sonuçlarına yönelik telefon aramalarını cevaplar.
- Kalite kontrol için kayıt sağlar.
- Laboratuvar faaliyetlerinin yönetimini denetler, örneğin teknisyenlerin verimliliğini izler.

Laboratuvar Bilgi Sisteminin (LIS)'in güvenli bulut dönüşümünün gerçekleştirilmesi durumunda, sağlık tesislerinde kullanılmakta olan cihazlar bulut ile çevrimiçi iletişimde olacak, cihazlar üzerinde çalışan tetkiklere ait sonuçlar bulut

üzerine aktarılabilecek, tetkik sonucuna etki edebilecek faktörleri izleyen sensörlerden gelen veriler bulut üzerinde işlenip, cihaz ve tetkik anomalileri gözlemlenebilir niteliğe sahip olacaktır. Bu dönüşüm ile geleneksel LIS sistemlerinde laboratuvar cihazlarının otomasyon iletişimi için pc gereksinimi yerine tıbbi cihaz-bulut ve sensör-bulut iletişimi için raspberry gibi küçük donanımlara bırakacaktır. Bu yazılımlar;

- Hastadan tahlil istenmesi,
- İstenen tahlile ilişkin numunenin alınması,
- Alınan numunenin barkodlanarak cihaza verilmesi,
- Verilen numuneye ilişkin olarak cihazın üretmiş olduğu sonucun laboratuvar teknisyenine, laboratuvar uzmanına, klinik doktoruna ve hastaya kadar ulaşması sürecini yönetmektedir.

2 Literatür Çalışması

Sağlık hizmeti sağlayıcıları, BT'nin sağladığı rekabet avantajını elde etmek için Bilgi Teknolojisini (BT) kullanmayı düşünmektedir. Bir Sağlık Bilgi Sistemi (HIS), sağlık hizmetlerini desteklemek için insanları, süreçleri ve teknolojileri içeren ve finansal, idari ve klinik yönleri içeren entegre bir sistemdir. Sağlık kurumlarında BT çözümlerini uygulamanın faydaları arasında tıbbi uygulamaları geliştirmek, karar verme süreçlerini desteklemek ve bilgi paylaşımını kolaylaştırmak yer almaktadır [5]. Sağlık sistemi; ilaç şirketleri, sigorta şirketleri, laboratuvarlar ve hastane tesislerinden oluşmaktadır. Tüm bileşenlerin devlet düzenlemelerine uyması ve hasta gizliliğini korurken verileri hızlı, doğru ve verimli bir şekilde paylaşması gerekir. Hasta verileri oldukça hassastır ve bunun tehlikeye girme riski, bulut bilişimin yavaş yavaş benimsenmesinin bir nedeni olarak kabul edilir. Sağlık sektöründe kritik sağlık bilgilerini yöneten herhangi bir yenilikçi teknoloji modeli hastanın mahremiyetini sağlamalıdır [6].

Sağlık sektörünün yaklaşık% 31'i şu anda bulut bilişim kullanmakta ve diğer sağlık hizmetleri kuruluşlarının çoğu bunu önümüzdeki beş yıl boyunca geçerli bir seçenek olarak görmektedir. Örgütlerin çoğunluğu hibrid modele geçmeden önce ilk olarak genel bulut modelini benimsemeyi ve nihayet uygunsa iş için özel bir bulutu kabul etmeyi amaçlamaktadır. Bulut modelleri ile ilerleyen ya da yakın gelecekte bunu yapmak isteyenler, finans modellerini uygulamakta yavaş olanlara göre finans ve verimlilik açısından daha avantajlı sayılmaktadır [7].

IoT (Nesnelerin İnterneti) sistemlerindeki sağlık uygulamaları son zamanlarda dikkat çekmeye başlamıştır, çünkü IoT sistemleri hastaların uzaktan izlenmesini kolaylaştıran birçok kullanışlı özellik sunmaktadır. Laboratuvar Bilgi Sistemlerinin modernizasyonu sürecinde medikal cihazlarla birlikte tetkik sonuçlarına etkisi olan dış etkenleri izleyen sensörlerin kullanılması ve bunların IoT adaptasyonunun sağlanması gerekliliği doğmuştur [8].

3 Cloud LIS Dönüşümü

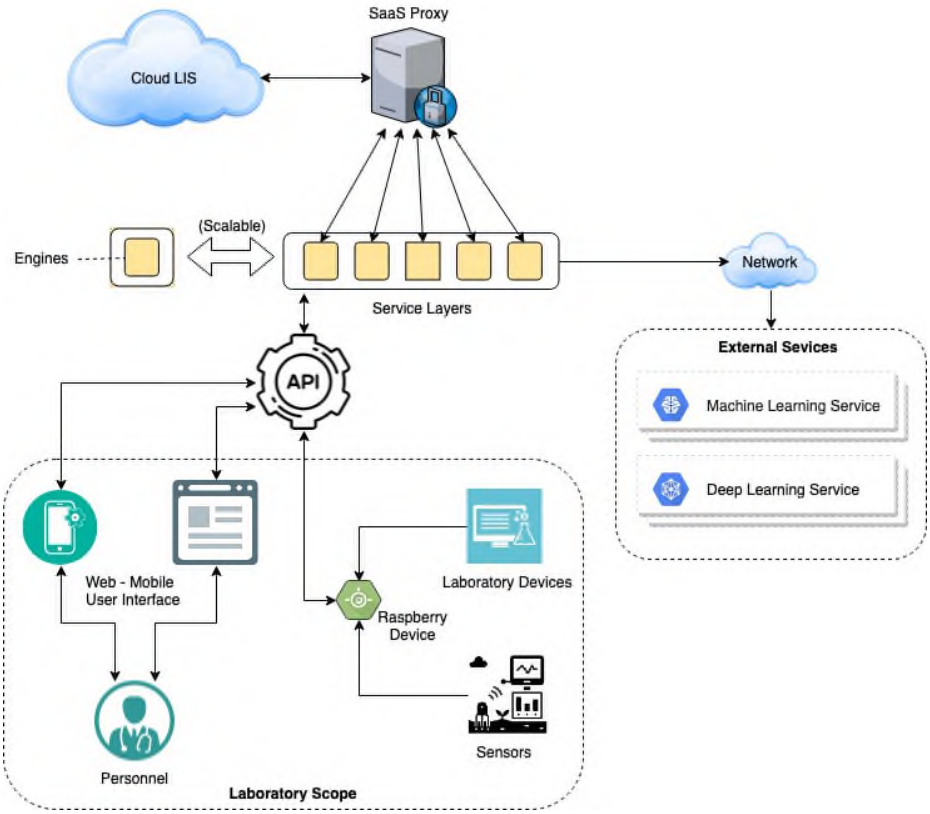
Bu çalışma kapsamında anlatılan Laboratuvar Bilgi Sistemi (LIS)'nin bulut dönüşümü mimari olarak Şekil 1'de gösterildiği gibi 4 katman ile gerçekleştirilebilir: ölçeklenebilir uygulama katmanı, API katmanı, medikal cihaz ve sensörlerin iletişim katmanı, mobil ve web arayüz katmanı.

Ölçeklenebilir (Scalable) uygulama katmanı. Sistemin eş zamanlı çalışabilmesi, çok fazla laboratuvarın cloud'a bağlanabilmesi ve artan işlem gücü miktarına uygun performans düşüklüğü yaşatmadan uyum sağlayan uygulama servisi tasarımına ihtiyaç bulunmaktadır. Büyük ölçekli bir kaynak yönetimi algoritması sağlama hedefi göz önüne alındığında kullanılması gereken algoritma iki katmana sahiptir. Dengeleme amacıyla üst katmanda proxy kullanılacaktır ve alt katmanda ise uygulama sunucuları katmanı kullanılacaktır. Buradaki amaç kaynak kullanımını maksimize etmektir [9].

API katmanı. Bu katman aracılığı ile uygulamanın metotlarını diğer uygulamaların kullanıma açarak uzaktan gelecek veri ve bilgi taleplerinin karşılanması amaçlanmaktadır. API katmanı, okunması ve kullanılmasının kolay olması sebebi ile JSON REST mimarisi ile oluşturulmalıdır. Kimlik doğrulama ve yetkilendirme kontrolleri bu katman üzerinde yapılmalı, işlemi doğrulanan ve onaylanan isteğe ait görevler uygulama katmanına aktarılmalıdır. Laboratuvardaki medikal cihazlar, sensörler, mobil - web uygulamaları bu katman üzerinden veri akışını yapmaktadır.

Medikal cihaz ve Sensörlerin iletişim katmanı. Geleneksel laboratuvar otomasyon sistemlerinde medikal cihazların tetkikleri çalışabilmesi ve üretmiş oldukları sonuçların sisteme aktarılabilmesi için masaüstü bilgisayar ihtiyacı bulunmaktadır. Laboratuvar Bilgi Sisteminin Cloud dönüşümü ile masaüstü bilgisayar ihtiyacı modernize edilerek, yerini raspberry gibi donanımlara bırakacaktır. Bu donanımlar üzerinde geliştirilmiş hafif uygulamalar sayesinde medikal cihaz - cloud, sensör - cloud online iletişimi sağlanmış olacaktır.

Mobil ve Web arayüz (User Interface) katmanı. Laboratuvar süreçlerinin yürütülebilmesi ve yönetilebilmesi için mobil ve web uygulamaları geliştirilecektir. Bu uygulamalar cloud ile API katmanı üzerinden iletişim sağlayacaktır. Mobil uygulama; laboratuvar numune alımı ve doğrulama işlemleri, laboratuvar uzmanına tetkik ve cihaz anomalilerini bildirimleri gibi laboratuvar süreçlerinin kalitesini arttırmayı amaçlamaktadır. Geliştirilecek Web uygulaması ile, yürütülen laboratuvar süreçlerinin tamamı, lokasyondan bağımsız olarak kullanılabilir niteliğe kavuşturulacaktır.



Şekil 1. Cloud LIS Mimarisi

4 Sonuçlar

Sağlık bilgi sistemlerinin en önemli bölümlerinden birisi olan Laboratuvar Bilgi Sistemi (LIS)'nin Cloud dönüşümü ile sağlık kurumlarının laboratuvar bölümlerinde teknolojik yenilik sağlanacak, çağın gereksinimlerine uygun kalitesi arttırılmış iş süreçlerine ve gelecekte daha etkin kullanım imkanı sağlayacak olan alt yapı elde edilecektir. Bu dönüşüm ile;

- Cloud'a bağlanan laboratuvarların ve cihazların takip ve izleme altyapısının oluşturulması,
- Sağlık bilgi sistemlerinde çağın ihtiyaç ve performans seviyelerine erişebilmesi,
- Laboratuvar test sonuçlarında doğruluk ve güvenilirlik artışı,
- Hasta numune alım ve sonuçlanma süreçlerinde kalite ve başarı artışı,
- Laboratuvar verimliliği ve raporlama kalitesinde artış,

- Laboratuvar cihazlarındaki anomalilerin tespit edilebilmesi, denetim ve uyarı mekanizmalarının oluşturulması,
- Mobil cihazlar ve uygulamalar ile Laboratuvarda görevli personelin Laboratuvar Bilgi Sistemi (LIS) ile etkileşiminin artması,
- Medikal ve Bilimsel çalışmalarda kullanılabilecek veri setlerinin oluşması,
- Makine öğrenmesi ve Derin öğrenme yöntemlerinde kullanılabilecek büyük veri setleri ve modellerinin oluşması,
- Kritik veri üreten laboratuvar cihazları, sensörler üzerindeki IoT dönüşümü ve etkileşimi

sağlanacaktır.

Laboratuvar Bilgi Sistemi'nin Cloud dönüşümü gerçekleştirilirken; Ölçeklenebilir uygulama katmanı, API katmanı, Medikal Cihaz - Sensör iletişim katmanları Node.js platformu üzerinde JavaScript dili kullanılarak geliştirilecektir. Mobil - Web uygulama katmanında ise Mobil uygulama için Android ve IOS işletim sistemlerine derleme imkanı sağlayan React Native framework'ü üzerinde JavaScript dili ile geliştirilecek, Web uygulaması için ise Angular framework üzerinde TypeScript dili kullanılarak geliştirme yapılacaktır.

Kaynakça

1. Artz, D. R., Sirintrapun, S. J.: Health Information Systems, Surgical Pathology 8 (2015) 255–268 http://dx.doi.org/10.1016/j.path.2015.02.014.
2. Sümen, E., Zayim N., Saka, O. : Laboratuar Bilgi Sistemi (LBS) Uygulaması: Benimseme ve Kullanıcı Memnuniyeti, 2. Ulusal Tıp Bilişimi Kongresi/Medical Informatics '05 Turkey, Pages: 102-107.
3. Henricks, W.H.: Laboratory Information Systems, Clin Lab Med 36 (2016) 1-11, http://dx.doi.org/10.1016/j.cll.2015.09.002.
4. Glandon, G. L., Smaltz, D. H. ve Slovensky, J. D. : Austin and Boxerman's Information Systems for Healthcare Management, 7th Edition (2008).
5. Alharbi, F., Atkins, A., Stainer, C., Al-Buti, H.A. : Strategic Value of Cloud Computing in Healthcare Organisations Using the Balanced Scorecard Approach: A Case Study from a Saudi Hospital, Volume 98, 2016, Pages 332-339.
6. Ali, O., Shrestha, A., Soar, J., Wamba, S.F.: Cloud computing-enabled healthcare opportunities, issues, and applications: A systematic review. International Journal of Information Management, Volume 43, December 2018, Pages 146-158.
7. Wan, D., Greenway, A., Harris, J. G., Alter, A. E.: Six questions every health industry executive should ask about cloud computing, http://www.accenture.com/SiteCollectionDocuments/PDF/Accenture_Cloud_Healthcare_PoV.pdf.
8. Woo, M. W., Lee, J., & Park, K. : A reliable IoT system for Personal Healthcare Devices, Volume 78, Issue P2, January 2018, Pages 626-640.
9. Iranpour, E., Sharifian, S., A distributed load balancing and admission control algorithm based on Fuzzy type-2 and Game theory for large-scale SaaS cloud architectures, Volume 86, September 2018, Pages 81-98.

Çevik Ontoloji Geliştirme 101: Yeniden Kullanım Tabanlı Çevik Bir Ontoloji Geliştirme Yöntemi

Cemil Abiş[1], Görkem Giray, Murat Osman Ünalır[1]

[1] Ege Üniversitesi, Bilgisayar Mühendisliği Bölümü, İzmir
91110019563@ogrenci.ege.edu.tr, gorkemgiray@gmail.com,
murat.osman.unalir@ege.edu.tr

Özet. Ontolojilerin bilgisayar bilimleri alanında yaygınlaşmasıyla birlikte, ontoloji geliştirme metodolojilerine yönelik çalışmalar da artmıştır. Genellikle ontoloji geliştirme deneyimlerinden ortaya çıkan bu metodolojiler, geliştirme süreçlerini farklı bakış açılarıyla ele almıştır. Ancak yazılım mühendisliğinde olduğu gibi olgunlaşmış ve kabul görmüş bir süreçten bahsetmek hala mümkün değildir. Bunun yanı sıra ontoloji geliştirme metodolojilerinin çözüm sağlayamadığı yeniden kullanım gibi bazı konular, ontoloji mühendisleri için önemli bir sorun olmaya devam etmektedir. Bu çalışmada mevcut yöntemlerin eksiklikleri dikkate alınarak, çevik ve yeniden kullanım tabanlı bir ontoloji geliştirme metodolojisi önerilmiştir.

Anahtar Kelimeler: Ontoloji, ontoloji geliştirme metodolojileri, ontoloji mühendisliği, ontoloji yeniden kullanımı, çeviklik.

Agile Ontology Development 101: A Reuse-based Agile Ontology Development Methodology

Cemil Abiş[1], Görkem Giray, Murat Osman Ünalır[1]

[1] Ege University, Department of Computer Engineering, Izmir
91110019563@ogrenci.ege.edu.tr, gorkemgiray@gmail.com,
murat.osman.unalir@ege.edu.tr

Abstract. With the spread of ontologies in the field of computer science, studies on ontology development methodologies have increased. These methodologies, which generally arise from ontology development experiences, have dealt with the development processes from different perspectives. However, it is still not possible to talk about a mature and generally accepted process as in software engineering. In addition, some issues, such as reuse, where ontology development methodologies cannot provide solutions, remain an important problem for ontology engineers. In this study, an agile and reuse-based ontology de-

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

velopment methodology is proposed considering the shortcomings of the existing methods.

Keywords: Ontology, ontology development methodologies, ontology engineering, ontology reuse, agility.

1 Giriş

Ontoloji geliştirme metodolojileri, ontoloji geliştirme sürecini kolaylaştıracak ve başarıyı arttıracak bir dizi faaliyeti içeren yönergelerdir. Bu süreçlerin bazıları doğrudan bilgi mühendisliğini temel almakta, bazıları ise yazılım geliştirme sürecinin kabul gören adımlarını ontoloji mühendisliğine göre uyarlamaktadır [1,2]. Ancak ontoloji geliştirme alanında, yazılım geliştirme metodolojilerindeki gibi kabul görmüş ve olgunlaşmış bir metodolojiden hala bahsedilememektedir.

Ontoloji geliştirme metodolojilerinin en önemli problemlerinden biri, mevcut ontoloji kaynaklarının yeniden kullanılmasıdır. Yeniden kullanım adımı birçok metodolojide bulunsa da bunun nasıl yapılacağı konusunda yeterince detaylı bilgi yer almamaktadır. Bunun yanı sıra yeniden kullanım sürecini destekleyecek araçların olmaması da önemli bir eksikliktir.

Bu çalışmada, mevcut metodolojilerin eksiklikleri değerlendirilerek, çevik ve yeniden kullanım tabanlı bir ontoloji geliştirme süreci önerilmiştir. Bildirinin ikinci bölümünde, bu konudaki temel kavramlar ve ilgili çalışmalara değinilecek ve eksiklikler üzerinde durulacaktır. Üçüncü bölümde önerilen yöntem detaylandırılacaktır. Dördüncü bölümde ise ontoloji geliştirme süreci ve yeniden kullanımdaki araçlar ve bu çalışma kapsamında geliştirilen uygulamadan bahsedilecektir. Son bölüm, değerlendirme ve gelecek çalışmalar hakkındaki bilgileri sunacaktır.

2 Temel Kavramlar ve İlgili Çalışmalar

Yeniden kullanım tabanlı ve çevik ontoloji geliştirme amacının açık olarak anlaşılabilmesi için mevcut ontoloji geliştirme yöntemlerinin, yeniden kullanımın ve çevikliğin ontoloji geliştirme bağlamındaki öneminin ayrı ayrı anlaşılması gereklidir. Bu nedenle, bu bölümde ilgili konular hakkında bilgiler verilecektir.

2.1 Ontoloji Geliştirme Metodolojileri

Ontolojilerin bilgisayar bilimlerinde yer edinmesi ve yaygınlaşmasıyla birlikte ontoloji geliştirme metodolojileriyle ilgili çalışmalar da hızlanmıştır. Genellikle geliştirilen ontolojilerden elde edilen deneyimleri esas alan yeni metodolojiler, ontoloji geliştirmenin başarıya ulaşmasını sağlayacak süreçleri ve aktiviteleri sunabilmeyi amaçlamıştır. Ontoloji geliştirme süreçlerine yönelik yöntemler, metodolojiler, teknolojiler ve araçlar üzerinde yapılan bu çalışmalar, Ontoloji Mühendisliği olarak adlandırılan yeni bir mühendislik disiplinini ortaya çıkarmıştır [3].

Grüninger ve Fox [4], TOVE projesi kapsamında geliştirilen ontolojiler sonucunda elde ettikleri deneyimlerle bir metodoloji önermişlerdir. Bu metodoloji, literatürde ontoloji geliştirme ile ilgili yer alan ilk çalışmalardan biridir. Ontoloji geliştirmenin ilk adımı olarak amaç ve kapsam çalışması yapılarak kullanıcı hikayelerine dayanan senaryolar oluşturulmaktadır. Bu adımı biçimsel olmayan yeterlilik soruları takip etmektedir. Terminoloji ve aksiyomlar, birinci derece mantık (first-order logic) kullanılarak biçimsel hale getirilmektedir. Çalışmada, geliştirme sürecinin yaşam döngüsü ile ilgili bir detay verilmemiş ve ontoloji yeniden kullanımından bahsedilmemiştir. Ayrıca aktiviteler ile ilgili detaylar yer almamaktadır.

Uschold ve King [5], Enterprise Ontology [6] isimli ontolojinin geliştirim sürecinde elde ettikleri deneyimler doğrultusunda bir süreç iskeleti önermiştir. Ana süreç, amaçların belirlenmesi, ontolojinin geliştirilmesi, değerlendirme ve dokümantasyon olmak üzere dört adımdan oluşmaktadır. Geliştirme aşamasında kavramsal modelin oluşturulması, bu modelin biçimsel bir dille kodlanması ve yeniden kullanım alt adımları yer almaktadır. Bu aktiviteler ile ilgili detaylı bilgi verilmemektedir. Ontoloji geliştirme süreci için bir yaşam döngüsü önerilmemiştir. Ayrıca, ontoloji yeniden kullanımından detay verilmeden bahsedilmektedir.

Methontology [1], önceki süreçlerden farklı olarak ontoloji geliştirme süreci için önerilen bir yaşam döngüsü üzerinde oluşturulmuştur. Prototipleme yaşam döngüsünü temel olan Methontology, içerdiği tüm aktiviteleri proje yönetimi, geliştirme ve destek olacak şekilde üç ana gruba ayırmıştır. Proje yönetimi, planlama, kontrol ve kalite yönetimi gibi aktiviteleri içermektedir. Geliştirme ile ilgili aktivitelerde, temel olarak kavramsal modelin oluşturulması ve ardından biçimsel modelin geliştirilmesi ile ilgili adımlar bulunmaktadır. Ontoloji yeniden kullanımı, destekleyici bir faaliyet olarak önerilmiştir ve ontoloji geliştirme sürecinin tüm adımlarında yer almaktadır.

Noy ve McGuinness, ontoloji geliştirme literatüründe yer alan önemli çalışmalardan biri olan Ontology Development 101'i önermişlerdir [2]. Bu çalışma, proje yönetimi ve destekleyici aktiviteleri de içeren kapsamlı bir süreçten çok geliştirme faaliyetlerine odaklanan bir çerçeve şeklindedir. Özellikle vurgulanan nokta, ontoloji geliştirme sürecinin doğası gereği yinelemeli olması gerektiğidir. Nesneye dayalı tasarımın bazı prensipleri de temel alınarak çeşitli aktiviteler önerilmiştir. Ontoloji yeniden kullanımına yer verilmiştir ve önemi vurgulanmıştır.

Diligent [7], ontoloji mühendisliğinin dağıtık, gevşek denetimli ve sürekli gelişen bir yapıda olması gerektiğini belirtmektedir. Bu yaklaşımda, ontolojinin bir başlangıç versiyonu tüm kullanıcılara sunulur ve kullanıcılar kendi yerelinde istediği değişikliği yapmakta özgürdür. Paylaşılan ontolojinin kalitesinden sorumlu olan merkezi bir yönetim bulunmaktadır. Bu yönetim, çekirdek ontolojide hangi değişikliklerin yapılacağına karar verir. Güncellemeler genellikle dağıtık bir şekilde çalışan kullanıcılar tarafından yapıldığından merkezi yönetim bu süreci gevşek bir şekilde yönetir. Kullanıcılar güncelleme yaptıkça da ontoloji gelişir. Ontoloji geliştirme süreci yinelemeli olarak gerçekleştirilir. Ancak yeniden kullanım ile ilgili bir detay yer almamaktadır.

Neon metodolojisi [8], diğer ontoloji geliştirme süreçlerinden farklı olarak katı bir şekilde tanımlanmış adımlara ve akışa sahip değildir. Bunun yerine ontoloji geliştirme için farklı akış alternatifleri sunmaktadır. Geliştirme süreçlerini ve özellikle yeniden kullanım gereksinimlerini dikkate alan dokuz farklı senaryo önerilmiştir. Neon, mev-

cut ontoloji ve diğer kaynakların yeniden kullanılması üzerine odaklanmış bir metodolojidir. Yeniden kullanım adımına sürecin temelinde yer verir ve diğer metodolojilerden farklı olarak detaylandırır. Ontoloji yaşam döngüsü, ontoloji mühendisinin tercihine bırakılmıştır.

Ontoloji süreçlerinin karşılaştırılması, bu çalışma kapsamındaki kriterler olan yaşam döngüsü ve yeniden kullanım üzerinden gerçekleştirilmiştir (Bkz. Tablo 1).

2.2 Ontoloji Yeniden Kullanımı

Bir problem alanı içerisinde yer alan mevcut ontoloji kaynaklarının yeniden kullanılması, ontoloji geliştirme sürecinin maliyetlerini azaltacak, yeni ontolojinin kalitesi arttıracak, bakımı kolaylaştıracak ve geliştirilen ontolojinin de yeniden kullanılabilir olmasını sağlayacaktır [9]. Ayrıca ilgili alan içerisindeki bilginin, birden fazla ontoloji ile temsil edilmesi engellenerek tutarlılık da sağlanmış olacaktır [10].

Ontoloji yeniden kullanımı, literatürde yer alan ve farkındalığı yüksek bir çalışma alanıdır. Ontoloji geliştirme süreçlerinin neredeyse tümü yeniden kullanımın önemini vurgulamaktadır. Ancak yeniden kullanımın, bir alt adım veya ayrı bir süreç olarak nasıl gerçekleştirileceğinden bahsedilmemektedir. Dolayısıyla ontoloji yeniden kullanımını tam olarak destekleyen bir yöntemden bahsedebilmek mümkün değildir.

Ontoloji yeniden kullanımı ile ilgili olarak son dönemde gerçekleştirilmiş olan bazı çalışmalar mevcut durumu ortaya çıkarmaya çalışmaktadır [10,11,12]. Bu çalışmaların ortaya koyduğu en önemli sonuç yeniden kullanımın ağırlıklı olarak ontolojinin bütün olarak dahil edilmesi (import) ile gerçekleştirildiğidir. Poveda, çalışmasında incelediği bağlı veri kaynaklarında, terimlerin %40 oranında mevcut kaynaklar üzerinden yeniden kullanım ile alındığını göstermiştir [12]. Bu terimlerin %67'si ise *owl:import* ile dahil edilmiştir. Başka bir ifade ile yer aldığı ontolojinin bütün olarak dahil edilmesi ile yeniden kullanılmıştır. Bu durum küçük ölçekli ontolojiler için kabul edilebilir olsa da büyük ölçekli ontolojiler için uygun değildir.

Ontoloji yeniden kullanımının istenilen noktada olmamasının en önemli nedeni yeniden kullanımı destekleyen süreç ve araçların eksikliğidir [10]. Hem süreç hem de araç kapsamında bütünsel yeniden kullanım ve parçalı yeniden kullanım yönlendirici bir şekilde ortaya konmalıdır. Ayrıca ontoloji yeniden kullanımının, mevcut gereksinimleri doğrudan etkileyen bir aktivite olduğu da göz önünde bulundurulmalıdır.

2.3 Ontoloji Geliştirme ve Çeviklik

2001 yılında yayınlanan Çevik Manifesto ile birlikte çevik yazılım geliştirme önem kazanmaya başlamış; Scrum ve Extreme Programming (XP) gibi birçok metodoloji yaygınlaşmıştır. Çevik süreçlerin genel prensibi geliştirmenin her aşamasında değişikliğe ayak uydurabilmeyi sağlamaktır. Bunu da yinelemeli ve artışlı yaşam döngüleri ile gerçekleştirir.

Değişime ayak uydurabilme yeteneği ontolojiler için de geçerli ve önemli bir özelliktir. Bilgi sürekli olarak güncellenmekte, mevcut ontoloji kaynaklarının kullanımı halihazırdaki gereksinimleri değiştirmektedir. Bu yüzden çevik yazılım geliştirme

Tablo 1. Ontoloji geliştirme metodolojilerinin karşılaştırılması.

Metodoloji	Proje Yönetimi	Yaşam Döngüsü	Yeniden Kullanım	Yeniden Kullanım Yaklaşımı	Araç Desteği/Kullanımı
Grüninger ve Fox	Belirtilmemiş	Belirtilmemiş	Belirtilmemiş	Belirtilmemiş	Belirtilmemiş
Uschold ve King	Belirtilmemiş	Belirtilmemiş	Var	Belirtilmemiş	Belirtilmemiş
Methontology	Var	Prototipleme	Var	Belirtilmemiş	Belirtilmemiş
Ontology Development 101	Belirtilmemiş	Yinelemeli	Var	Belirtilmemiş	Protege
Diligent	Belirtilmemiş	Yinelemeli	Belirtilmemiş	Belirtilmemiş	Belirtilmemiş
Neon	Belirtilmemiş	Şelale/Yinelemeli	Var	Detaylı	Neon Toolkit

süreçlerinin başarısını ontoloji geliştirme süreçlerine uyarlamak önemli bir katkı olacaktır. [13] ve [14] çalışmalarında çevik ontoloji geliştirme üzerine farklı öneriler ve yöntemler sunulmuştur. Bu çalışmanın mevcut çalışmalardan farkı, yeniden kullanımı da çevik bir bakış açısı ile sürecin bütününe dahil etmek olacaktır.

3 Yeniden Kullanım Tabanlı Çevik Ontoloji Geliştirme Yöntemi

Mevcut ontoloji geliştirme metodolojileri ve ontoloji mühendisliğinin kapsamı incelendiğinde başarılı bir ontoloji geliştirme metodolojisinin bazı temel özelliklere sahip olması gerekliliği ortaya çıkmaktadır. Bu çalışma kapsamında iki temel özellik üzerinde durulmaktadır: Yinelemeli yaşam döngüsü ve yeniden kullanım.

Ontolojilerin bilgisayar bilimlerindeki kullanımı, bir problem alanındaki bilginin biçimsel olarak modellenmesini içerir. Bu biçimsel model içerisinde kavramlar, bu kavramların özellikleri ve kavramlar arasındaki ilişkiler açık bir şekilde tanımlanmıştır [15]. Ontolojilerin modellediği bilginin sürekli güncellenebilme potansiyeli ve geliştirme sırasında ortaya çıkabilecek ve gereksinimleri tümüyle etkileyebilecek değişiklikler, ontoloji geliştirmenin yinelemeli bir yaşam döngüsü üzerine kurgulanmasını gerektirir. Tablo 1'deki karşılaştırma özetinde de görülebildiği gibi ontoloji geliştirmenin erken dönemlerinde önerilen çalışmalar dışındaki metodolojiler, yinelemeli veya prototipleme tabanlı yaşam döngülerini temel almaktadır.

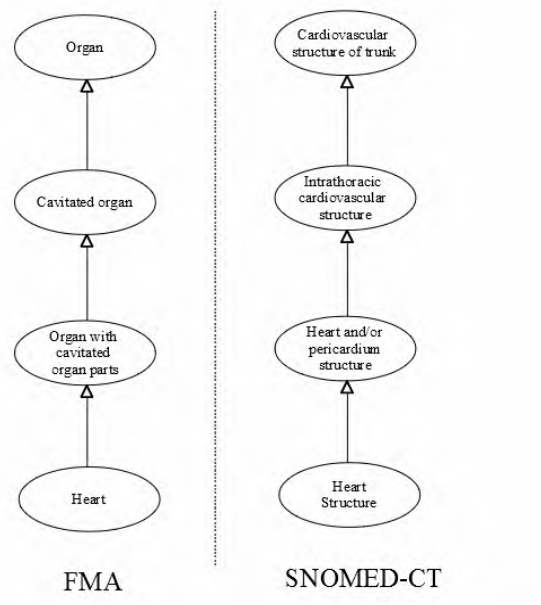
Ontoloji mühendisliği, birçok mühendislik disiplininde olduğu gibi yeniden kullanımı önemser. Ancak ontoloji yeniden kullanımı karmaşık bir alt süreç olarak karşımıza çıkar. Mevcut ontolojilere erişimin sağlanması, değerlendirme, modülerleştirme, bütünleştirme ve daha birçok konu bu alt sürecin parçasıdır.

Ontoloji yeniden kullanımının yazılım mühendisliğinde karşımıza çıkan yeniden kullanımdan en önemli farkı, süreci bütünüyle etkileyebilmesidir. Ontoloji yeniden kullanımını doğru konumlandırabilmek için bu noktanın anlaşılabilmesi önemlidir. Örneğin; bir ontoloji mühendisi sağlık alanında bir ontoloji geliştiriyor olabilir. Bu alanda mevcut ontoloji kaynaklarına ulaşabilmek oldukça kolaydır. FMA[1] ve SNOMED-CT[2] gibi literatürde kabul görmüş birçok ontoloji yer almaktadır. Ontoloji mühendisi, organları ifade edecek kavramları yeniden tanımlama maliyetlerinden kurtulmak ve birlikte çalışabilirliği sağlamak için bu ontolojilere başvurabilir. Ancak ontoloji seçimi, kavramın yeniden kullanım düzeyi (tekil olarak veya tüm üst sınıfları ile birlikte) ve yeni ontolojiye nasıl bütünleştirilebileceği gibi birçok konu ontoloji geliştirme sürecinin başında tanımlanmış olan gereksinimleri bütünüyle etkileyecektir. Bu farklılığı açıklayabilmek için *Heart (Kalp)* kavramının FMA ve SNOMED-CT ontolojilerindeki tanımı Şekil 1'de üst sınıf ilişkileriyle birlikte gösterilmiştir. Görüldüğü gibi aynı kavram iki ontolojide tamamen farklı şekilde tanımlanmıştır.

Ontoloji mühendisi, SNOMED-CT içerisindeki tanımı tercih ederse bu kavramın tanımıyla birlikte diğer kavramlar da yeni ontolojiye dahil olacaktır. FMA tercihinde

[1] http://si.washington.edu/projects/fma
[2] http://www.snomed.org/

de benzer durum söz konusudur. Hangi ontoloji seçilirse, ontoloji geliştirmenin gereksinimleri, yeterlilik soruları ve kapsam o ontolojiye göre güncellenebilecektir. Bu etkileşim, yinelemeli/artışlı ve yeniden kullanımı detaylı bir alt süreç olarak dahil eden bir metodoloji ihtiyacını doğurmaktadır. Bu çalışmada, çevik yazılım geliştirme prensiplerinin başarısı da dikkate alınarak yeniden kullanım tabanlı çevik bir ontoloji geliştirme süreci önerilmektedir.



Şekil 1. *Kalp* kavramının FMA ve SNOMED-CT ontolojilerindeki tanımı[3].

Çevik Ontoloji Geliştirme 101 süreci, biri ana yineleme olmak üzere dört alt süreçten oluşmaktadır: (1) Kapsam ve gereksinim analizi, (2) geliştirme, (3) yeniden kullanım ve (4) modüler yeniden kullanım (Bkz. Şekil 2). Her bir alt süreçteki adımlar birden fazla kez yinelenebilir. Yineleme tamamlandıktan sonra, ilgili bağlantı noktası üzerinden üst sürece dönüş yapılır ve yineleme üst süreçte devam eder.

Çevik Ontoloji Geliştirme 101, bir çerçevedir. Proje yönetimi ile ilgili detaylara yer verilmemiştir. Her bir aktivite ontoloji mühendisleri için birer öneridir. Aktivitelerin nasıl hayata geçirileceği ontoloji mühendisleri ve proje yöneticilerine bırakılmaktadır. Bu yaklaşım, çevik metodolojilerle de uygunluk göstermektedir.

3.1 Kapsam ve Gereksinim Analizi

Kapsam ve gereksinim analizi, geliştirme yinelemesinin planlama aktivitesinin detayına karşılık gelen bir alt süreç olarak konumlandırılmıştır. Projenin ilk adımı olarak

[3] Şekil üzerinde yer alan kavramların, FMA ve SNOMED-CT ontolojilerindeki İngilizce tekil isimleri kullanılmıştır.

ve her geliştirme yinelemesinden hemen önce işletilir. Projenin başlangıç aşamasında işletilmesi durumunda, bu alt sürecin faaliyetleri proje bazlı olarak daha detaylı gerçekleştirilecektir.

Kapsam ve gereksinim analizi, Ontoloji Geliştirme 101 yönergesinin Alan ve Kapsamın Belirlenmesi adımını kapsamaktadır [2]. Diğer metodolojilerdeki planlama ve kapsam analizi dahilindeki aktivitelere karşılık gelmektedir. Bu yinelemenin alt adımları aşağıdaki gibidir:

- **Gereksinim analizi:** Ontolojinin içereceği problem alanı, kapsamı, hedef kullanıcıların kim olduğu gibi sorulara cevap alınan adımdır. Ağırlıklı olarak proje başlangıcında işletilecek olmasına rağmen süreç boyunca ortaya çıkabilecek değişiklik taleplerine karşı her bir geliştirme yinelemesinden önce tekrar edilir.
- **Yeterlilik sorularının belirlenmesi:** Yeterlilik soruları, bir ontolojinin cevaplaması gereken soruları ifade eder [2]. Kullanıcı hikayeleri gibi kolaylaştırıcı yaklaşımlar uygulanabilir. Yeterlilik soruları, ontolojinin geçerliliğinin doğrulanabilmesi için birer test kriteridir. Yeterlilik soruları, [13] çalışmasında belirtilen örneğe benzer şekilde Tablo 2'de gösterildiği gibi oluşturulabilir.

Tablo 2. Yeterlilik sorusu örneği [13].

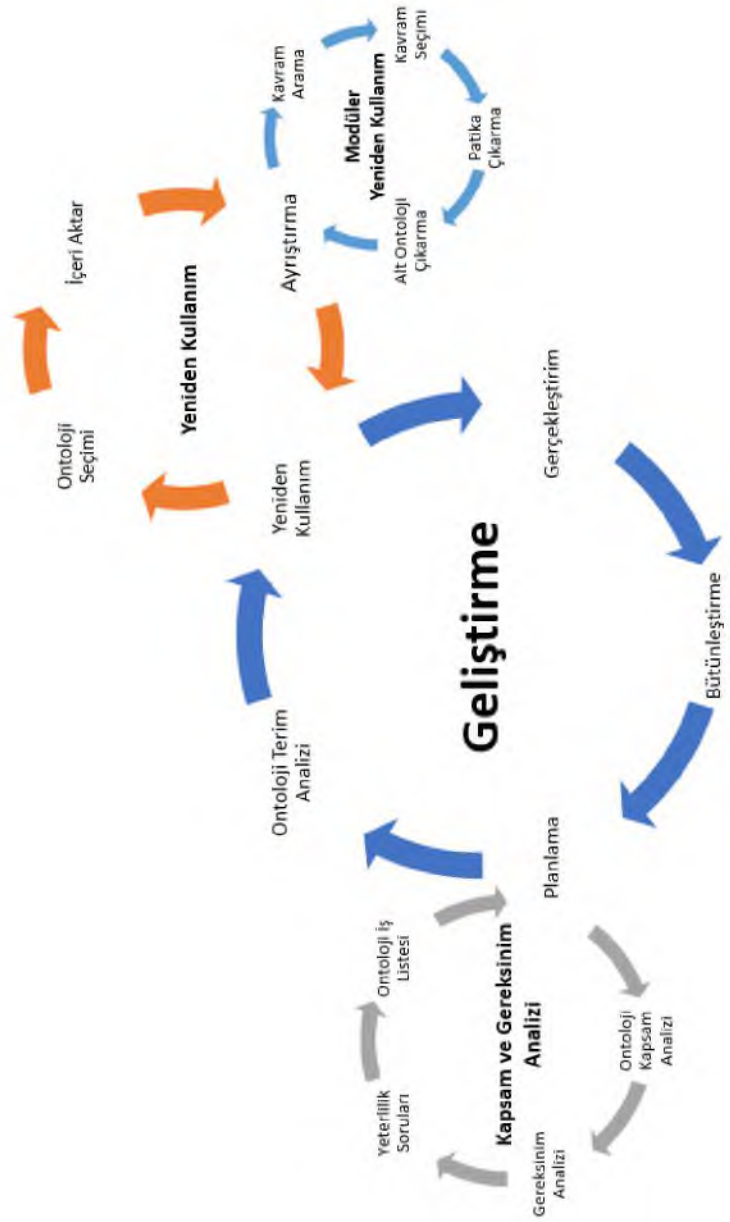
Tanımlayıcı	5
Soru	Taşıt tipindeki kavramlar nelerdir?
Cevap	Kara, deniz ve hava taşımacılığında kullanılan tüm taşıtlar
Örnek	Araba, gemi, uçak, tren vb.
Bağlı olduğu sorular	1, 2

- **Ontoloji iş listesi:** Ontoloji iş listesi, Scrum gibi çevik yazılım geliştirme metodolojilerinde kullanılan *Ürün İş Listesi (Product Backlog)* tanımına karşılık gelmektedir. Ontoloji iş listesi, yeterlilik sorularını birer iş elemanı olarak barındırmaktadır. Her bir geliştirme yinelemesi, hangi sorulara cevap vermesi gerektiğini planlama aşamasında belirlemelidir.

3.2 Geliştirme

Geliştirme yinelemesi, ontoloji geliştirmenin ana sürecidir. Diğer tüm yinelemeler, geliştirmeye bağlı olarak gerçekleştirilir. Bu sürecin çıktısı olarak bir ontoloji üretilir ve bu ontoloji yeterlilik soruları ile doğrulanabilir. Ontoloji Geliştirme 101 yönergesinin alan ve kapsam analizi dışındaki diğer tüm adımları, bu süreç içerisinde gerçekleştirilmektedir [2].

- **Planlama:** Geliştirme sürecinin ilk adımıdır. Kapsam ve gereksinim analizini, alt süreç olarak işletir ve yineleme planlamasını gerçekleştirir. Planlama sonrasında, ilgili yinelemede cevaplanacak yeterlilik soruları belirlenmelidir.



Şekil 2. Çevik Ontoloji Geliştirme 101 süreci.

- **Ontoloji terim analizi:** Belirlenen yeterlilik soruları kapsamında, ilgili yineleme içerisinde modellemesi ve gerçekleştirimi yapılacak terimlerin belirlenmesi faaliyetlerini içermektedir. Ontoloji Geliştirme 101, bu adımı *önemli terimlerin listelenmesi* olarak belirtir ve sürecin üçüncü adımı olarak yer verir [2].

- **Ontoloji yeniden kullanımı:** Yeniden kullanılacak ontolojilerin belirlenmesi, kapsam analizi sırasında kısmi olarak gerçekleştirilebilir. Ancak yeterlilik soruları ve terimler nihai olarak hangi ontoloji kaynaklarının yeniden kullanıma aday olduğunu belirleyebilmek için önemlidir. Ontoloji Geliştirme 101, yeniden kullanıma, terimlerin belirlenmesinden önce ikinci adım olarak yer vermektedir [2]. Diğer metodolojilerde de yeniden kullanım adımı ağırlıklı olarak geliştirmenin öncesinde yer bulmaktadır. Yeniden kullanım, kendi içerisinde bir alt süreç olarak işletilir.
- **Gerçekleştirim:** Bu adım sonrasında biçimsel bir dille (OWL, RDF/S vb.) ifade edilmiş bir ontoloji parçası ortaya çıkmalıdır. Kavramsal modelleme gibi aktiviteler uygulanabilir. Geliştirme adımının nasıl işletileceği ontoloji mühendisinin tercihine bırakılmıştır. Ontoloji Geliştirme 101, bu adıma karşılık dört alt adım içermektedir [2].
- **Bütünleştirme:** Gerçekleştirim adımının çıktısı olan ontoloji parçası, önceki yinelemelerde elde edilmiş ontoloji ile bütünleştirilir.

3.3 Yeniden Kullanım

Ontoloji yeniden kullanımı tüm ontoloji geliştirme sürecini etkileyen ve değiştirebilen bir alt süreçtir. Dolayısıyla geliştirmenin bir parçası olarak ve yinelemeli bir biçimde gerçekleştirilmelidir. Ontoloji yeniden kullanımı sadece süreç olarak değil araç olarak da desteklenmelidir. Ontoloji Geliştirme 101, bu adımı ikinci aktivite olarak tanımlamaktadır [2]. Bu alt sürecin adımları aşağıda ifade edilmiştir:

- **Ontoloji seçimi:** Gereksinim analizi, yeterlilik soruları ve belirlenen terimler yeniden kullanılabilecek ontolojilerin seçimi için birer referanstır. Bioportal gibi ontoloji depoları, ontoloji arama motorları ve DBpedia gibi bağımsız ontoloji kaynakları hedef ontolojilerin bulunabilmesi için birer kaynaktır.
- **İçeri aktarma (import):** FOAF, BFO ve Dublin Core gibi küçük ölçekli ontolojiler doğrudan *owl:import* ile yeni ontolojiye bütünsel olarak dahil edilebilir.
- **Ayrıştırma**: DBpedia, FMA ve SNOMED-CT gibi büyük ölçekli ontolojilerin bütün olarak içeri aktarılabilmesi performans ve uygulanabilirlik açısından mümkün değildir. Bu yüzden bu tip ontolojiler içerisinden yeniden kullanımın parçalı olarak yapılabilmesini sağlamak gerekmektedir. Bu adım yeniden kullanımın bir alt süreci olarak yinelemeli biçimde gerçekleştirilebilir.

3.4 Modüler Yeniden Kullanım

Büyük ölçekli ontolojilerden yeni ontoloji içerisine dahil edilecek parçaların oluşturulması, kendi içerisinde yinelemeli bir alt süreç olarak uygulanmaktadır. Bu alt süreç ontoloji mühendisi ile etkileşimli bir biçimde ve araç destekleri ile yarı-otomatik bir biçimde gerçekleştirilmelidir. Ontoloji mühendisi, kavramları tekil olarak, patikalar halinde veya ontoloji modülleri biçiminde yeni ontolojiye dahil edebilmelidir. Bu alt süreç, ontoloji geliştirme sürecinin en karmaşık adımıdır. Bu adımı destekleyecek araç ve uygulama daha önceki çalışmalarımızda geliştirilerek ortaya konulmuştur [16]. Modüler yeniden kullanımın alt adımları aşağıdaki gibidir:

- **Kavram arama:** Ontoloji mühendisi, yeniden kullanılmak üzere belirlenen ontolojiler üzerinde ihtiyaç dahilindeki kavramları bulabilmek için arama yapar.
- **Kavram seçimi:** Yeni ontolojiye tekil olarak dahil edilmek istenen kavramlar seçilerek aktarılmalıdır.
- **Patika çıkarma:** Yeniden kullanılan ontolojiden seçilen ve ayrık olarak görülen iki kavram arasında patikalar olabilir. Ontoloji mühendisi, bir araç yardımıyla bu patikaları açığa çıkarabilmeli ve patika üzerindeki diğer kavramları da ontolojiye ekleyebilmelidir.
- **Alt ontoloji çıkarma:** Birbiri ile olan bağları açıkça görünmeyen ikiden fazla kavramı içeren ontoloji modülleri, ontoloji mühendisinin desteğiyle bir araç üzerinden yarı otomatik olarak elde edilebilmelidir. Böylelikle, büyük ontolojilerin içerisinden ilgi dahilindeki alt bölümler elde edilerek yeni ontolojiye eklenmiş olacaktır. Patika ve alt ontoloji çıkarma, [16] çalışmamızda önerilen yöntemlerle gerçekleştirilebilir.

4 Ontoloji Geliştirme Araçları

Ontoloji geliştirme, alt adımlardan oluşan karmaşık bir süreçtir ve yazılım araçlarıyla desteklenmesi başarı için gereklidir. Geliştirme süreci boyunca farklı araçlardan faydalanılabilir. Planlama için proje yönetim araçları tercih edilebilir. Ayrıca kelime işlemciler, elektronik tablo yazılımları ve modelleme araçları ontolojinin geliştirme aşamalarında ontoloji mühendisleri tarafından kullanılabilir.

Ontolojinin biçimsel olarak gerçekleştirimi için Protege ve benzeri ontoloji geliştirme araçlarına ihtiyaç duyulacaktır. En kritik nokta, yeniden kullanımın ifade edilmiş tüm detaylarını yarı-otomatik bir şekilde destekleyebilecek bir yeniden kullanım aracı olacaktır. Bu eksiklik konusunda, önerilen çevik ontoloji geliştirme sürecini desteklemek amacıyla bulut tabanlı bir yeniden kullanım aracı geliştirilmiştir. Bu araç ontoloji kaynakları üzerinde patika bulma ve alt ontoloji çıkarma gibi karmaşık süreçleri barındıran sunucu bileşenleri ve bu bileşenlere ait servisleri kullanıcılara sunan Protege eklentisinden oluşmaktadır. Araç ile ilgili detaylı bilgilere [16] çalışmasında yer verilmiştir.

5 Sonuç ve Gelecek Çalışmalar

Bu çalışmada, mevcut ontoloji geliştirme metodolojilerinin eksikliklerini gidermek ve ontoloji geliştirme sürecini kolaylaştırabilmek amacıyla çevik bir ontoloji geliştirme süreci önerilmiştir. Bu sürecin en önemli noktası yeniden kullanımı yinelemeli bir alt süreç olarak içermesi ve detaylandırmasıdır.

Gelecek çalışmalar arasında yöntemin bir durum senaryosu ile doğrulanması yer almaktadır. Bu amaca yönelik yeni bir ontoloji geliştirmek yerine Sleep Domain Ontology (SDO) [17] gibi yeniden kullanımı etkin bir şekilde uygulamış hazır bir ontolojinin tersine mühendislik ile yeniden geliştirilmesi amaçlanmaktadır.

İlerleyen dönemlerde, süreci desteklemek amacıyla geliştirilmiş ontoloji yeniden kullanım eklentisinde de yenilikler yapılması planlanmaktadır. Bu yeniliklerin başın-

da ontoloji değerlendirme yöntemlerin bütünleştirilmesi gelmektedir. Ayrıca yeniden kullanım sürecinin kullanıcı tercihlerine göre özelleştirilebilmesi de sağlanacaktır.

Kaynakça

1. Fernandez-Lopez, M. et al.: METHONTOLOGY: from Ontological Art towards Ontological Engineering. In: Proceedings of the AAAI97 Spring Symposium. pp. 33--40, Stanford, USA (1997).
2. Noy, N.F., McGuinness, D.L.: Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory. 25 (2001).
3. Suárez-Figueroa, M.C., García-Castro, R., Villazón-Terrazas, B., Gómez-Pérez, A.: Essentials in Ontology Engineering: Methodologies, languages and tools. Proc. of the 2nd Workshop on eeBuildings Data Models. 9-21 (2011).
4. Gruninger, M., Fox, M.S.: Methodology for the Design and Evaluation of Ontologies. In: IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing, 1–10 (1995).
5. Uschold, M., King, M.: Towards a Methodology for Building Ontologies. In: Methodology. pp. 275–280 (1995).
6. Uschold, M., King, M., House, S.B.R., Moralee, S., Zorgios, Y.: The Enterprise Ontology. The Knowledge Engineering Review. 13, 31–89 (1995).
7. Pinto, H.S., Staab, S., Tempich, C.: DILIGENT: Towards a fine-grained methodology for DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies. In: 16Th European Conference on Artificial Intelligence - Ecai. pp. 393–397 (2004).
8. Suarez-Figueroa, M.C., Gomez-Perez, A.: NeOn Methodology for Building Ontology Networks: a Scenario-based Methodology. In: on SOFTWARE, S.& S.T. (S3T 2009) I.C. (ed.) Proceedings of the International Conference on SOFTWARE, SERVICES & SEMANTIC TECHNOLOGIES (S3T 2009). (2009).
9. Kendall, E.F., McGuinness, D.L.: Ontology Engineering, Morgan & Claypool Publishers (2019).
10. Kamdar, M.R., Tudorache, T., Musen, M.A.: A Systematic Analysis of Term Reuse and Term Overlap across Biomedical Ontologies. Semantic Web. 0, 1–20 (2016).
11. Ochs, C., Perl, Y., Geller, J., Arabandi, S., Tudorache, T., Musen, M.A.: An empirical analysis of ontology reuse in BioPortal. Journal of Biomedical Informatics. 71, 165–177 (2017).
12. Villalón, M.P., Suárez-Figueroa, M.C., Gómez-Pérez, A.: The Landscape of Ontology Reuse in Linked Data. Proceedings Ontology Engineering in a Data-driven World (OEDW 2012). (2012).
13. Peroni, S.: A simplified agile methodology for ontology development. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 10161 LNCS, 55–69 (2017).
14. Keet, C.M., Lawrynowicz, A.: Test-Driven Development of Ontologies. In: Sack, H. et al. (eds.) ESWC. pp. 642–657 Springer (2016).
15. Guarino, N. et al.: What is an Ontology? In: Handbook on ontologies. pp. 1--17 Springer (2009).
16. Abiş, C., Giray, G., Ünalır, M. O., Ontoloji Yeniden Kullanimi icin Bulut Tabanli Bir Yazilim Mimarisinin Gerceklestirilmesi, Ulusal Yazılım Mühendisliği Sempozyumu (2018).
17. S. Arabandi, C. Ogbuji, S. Redline, R. Chervin, J. Boero, R. Benca et al., "Developing a Sleep Domain Ontology," in Proc. AMIA Clinical Research Informatics Summit, (2010).

Otomotiv Sektöründe Genişletilmiş Çevik Yazılım Geliştirme Süreci Deneyimi

Fatih Öztürk[1], Koray Bilgin[1], Cem Sezer[1], Seda Güneş[1]
[1] Red Pine Software R&D Centre,
İzmir, Turkey
{fatih.ozturk, koray.bilgin, cem.sezer, seda.gunes}@redpine.com.tr

Özet. Otomotiv sektörü için geliştirilen gömülü yazılımların karmaşıklığı İleri Sürücü Destek Sistemleri'ndeki (ADAS) müşteri gereksinimlerine uygun olarak hızla artmaktadır. Artan gereksinimleri karşılamak amacıyla çevik yazılımı genişleten altyapılar otomotiv sektörüne uyarlanmıştır. Bu da büyük projelerdeki çıktıların daha sık ve öngörülebilir olmasını sağlamıştır. Ancak, dağıtık takımlar için kritik güvenlik sistemlerini endüstriyel ölçekte çeviklikle geliştirmek epey zorlayıcıdır. Bu çalışmada, genişletilmiş çevik yazılım süreci tabanlı bir küresel organizasyonda fonksiyonel güvenlik için V-Modeli uygulamasında deneyimlediğimiz zorluklar ve kilit noktalar anlatılmıştır.

Anahtar sözcükler: SAFe, ADAS, V-Modeli, Gömülü Yazılım, Yazılım Süreç İyileştirme, Çevik Yöntemler

Scaled Agile Software Development Process Experience in Automotive

Fatih Öztürk[1], Koray Bilgin[1], Cem Sezer[1], Seda Güneş[1]
[1] Red Pine Software R&D Centre,
İzmir, Turkey
{fatih.ozturk, koray.bilgin, cem.sezer, seda.gunes}@redpine.com.tr

Abstract. The complexity of embedded software in automotive industry is rapidly increasing in parallel with the customer requirements in Advanced Driver Assistance Systems (ADAS). In order to fulfil the emerging requirements, Agile Software scaling frameworks have been adopted to automotive industry. This provided better frequency and predictability of releases in large projects. However, developing safety-critical systems with industry-scale agility is still a big challenge for distributed teams. In this study, we have identified the difficulties and key points that we experienced while applying V-Model for functional safety in a Scaled Agile driven global organization.

Keywords: SAFe, ADAS, V-model, Embedded Software, Software Process Improvements, Agile Methodologies

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

1. Introduction

In recent years, the use of various Agile methods in different organisations has been increased, while the use of Waterfall methodology is decreasing [17]. Previous studies have indicated the successful adoption of Agile methods in embedded software projects [18] together with others. The continuous development and test-driven mindset characteristics of embedded software development are compatible with the Agile principles.

Agile methodologies provide flexibility for the engineering teams especially in large-scale embedded projects. Scrum is the most preferred method among the various Agile methods such as XP (Extreme Programming), Kanban and Feature Driven Development (FDD). Scrum method does not describe reporting, source control or coding but it provides better predictability and risk management by using iterative and repetitive models.

Nowadays, Advanced Driver Assistance Systems (ADAS) are getting more critical as more applications are expected on premium vehicles including various forms of autonomous drive. Some safety functions are also becoming mandatory like Electronic Braking System (EBS), increasing the demand for all automotive range. Safety critical functionality is the major factor in complexity of the ADAS application development. In order to provide high-quality software components with the safety critical functionality, some specific procedures are required. In addition to the functional requirements safety requirements must also be fulfilled as described in the Functional Safety standard (ISO26262) [19].

The software in automotive industry is usually developed by multiple suppliers for single OEM (Original Equipment Manufacturer). V-Model is commonly adopted to automotive software development cycle. Suppliers are involved in the development process with a wide range of responsibilities such as application development, integration and testing. OEMs usually handle requirement specification, system design and final verification with acceptance testing.

The aim of this study is to give an insight and make a comparison for the use of Agile methodologies in the automotive software development process. In Section 2, some related work is available on applying Scaled Agile Framework (SAFe) practices for developing safety critical applications on AUTOSAR framework. In Section 3, background in automotive sector regarding Automotive SPICE, V-Model and ISO26262 are provided. In Section 4, our software development process experience with SAFe is presented. In Section 5 conclusion and future work are provided.

2. Related Work

Scaled Agile aspects and principles have been examined and presented [1] for some years now. Variety of Agile frameworks have been proposed [3] since the first models for scaling Agile in organizations was practised [2]. The common goal of these Agile frameworks is to increase predictability of the releases and the visibility across the whole organization. In order to achieve this goal in the entire enterprise, Agile mind-set has been applied in different disciplines, such as Architecting, Design, Marketing, Portfolio Management or Program Management rather than only using it in software development team level [1].

Among Agile frameworks, SAFe has become the most popular and the most adopted one [4] for multi-site organizations who are scaling their Agile development processes. The growing trend for globally distributed companies to adopt Agile methods also emerged SAFe practises in global organizations [5].

In automotive industry, manufacturers and suppliers have been implementing complex applications for more than a decade, such as ADAS applications [7]. They produce safety critical software by following AUTOSAR specifications as an open standard for automotive electronics. The AUTOSAR standard provides modularity, scalability and reusability of functions. However, there is no formal verification of reliability and safety concerns in AUTOSAR. Car manufacturers address these concerns by building compliance to ISO26262 [19] in accordance to the Automotive SPICE model [22]. Plan-Driven Methods were used in early applications of the protocols in the automotive industry. Automotive manufacturers also adopted Agile frameworks to their development processes [20], although the conflicts between plan-driven processes and Agile approaches emerged in some early studies [11], [12], [13], [14].

Technological improvements rapidly transformed modern cars from mechanical entities to complex embedded systems. Industry players have continuously improved software engineering technologies to stay ahead of the game, as well as improving their underlying processes and practices. High-end cars currently have around 100 electronic control units (ECUs), running multi-million lines of software spanning across safety-critical, driver assistance, comfort, and entertainment related applications [8]. Increasing complexity in software functionality emerged the need of distributed software and distributed development environments as the characteristics of automotive industry [9].

Agile practises and continuous delivery of software brought lots of advantages to automotive industry as well as in the embedded software world, although automakers lately started deploying Agile development. The biggest advantage is the efficiency in software development while maintaining the centralized decision-making necessary at the enterprise level. As the software complexity in the ECUs increase, there is need to address more sophisticated problems. SAFe provides the environment to mitigate these problems with a coordinated strategy for large-scale and complex projects with teams that number into the hundreds.

The key challenges to Agile adoption in the automotive industry involved transforming the organizational structure and culture, achieving a shorter release cycle without compromising quality [10]. Having this in mind, the paper presents a set of challenges relevant when Agile practices are scaled beyond a single team in organisations developing automotive software.

3. Background in Automotive Software

ADAS projects consist of complex software components and an underlying framework running on several ECUs. These components or framework can also be used in different car manufacturer projects. As depicted in Figure 1, ECU software system consists of more than one sub system aiming to serve different functionality such as lifecycle service and watchdog service. These services are considered as AUTOSAR compliant SWCs (Software Components) which are separated from the AUTOSAR basic

software stack [23] providing a layered software architecture via RTE (Run Time Environment) interface.

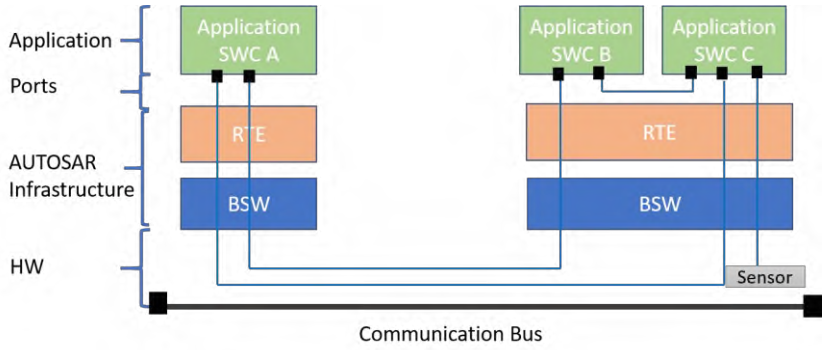


Fig. 1. The AUTOSAR Layered Architecture

AUTOSAR stack brings us a layered architecture along with the requirements of the ADAS framework and having cross functional SWCs on different hosts with different ASIL levels defined in ISO26262 [21]. After collecting the requirements, system engineers have the responsibility to design the system so that hardware and software systems are compatible.

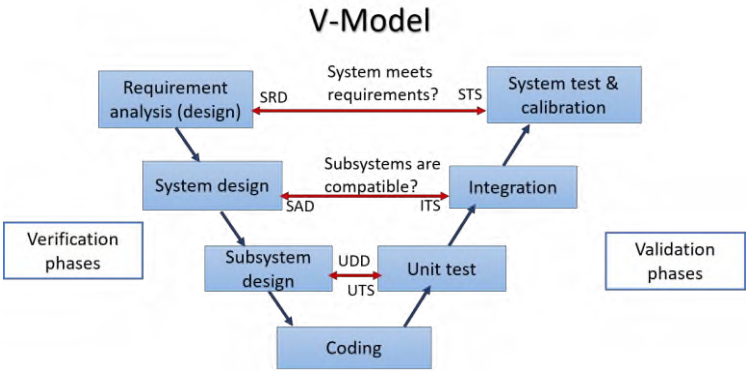


Fig. 2. Illustration of verification techniques within V-Model

4 Scaled Agile Software Lifecycle Experience in Automotive

Heterogeneous and distributed software development on many automobile variants and configurations is the main characteristic of automotive industry. In the following three subsections we present some of the difficulties that we experience as a contributor company in this distributed environment. Subsection 1 and Subsection 2 give examples of improvements initiated by the sprint retrospective meetings which mitigated the negative effects on team's performance. Subsection 3 presents some actions taken as a result of the Problem-solving workshops in order to identify the ways to improve the team performance in the Program level of SAFe.

4.1. Agility in different levels of V-Model

There are several documents which are created and maintained during the development cycle according to the V-model process. These documents are linked to each other from any subsections to provide a chain from requirement specification to validation procedure. After capturing all the functional and non-functional requirements, SRD (Software Requirement Document) is created in Requirement Analysis Phase. These requirements are analysed further to design SWC functionality in unit level which is translated into SAD (Software Architecture Document) in System Design Phase. UDDs (Unit Design Document) are constructed at the same time as the developers write the code by following the SAD which is created in Subsystem Design Phase. Doxygen comments, function prototypes and global variable definitions and unit types are added in UDDs. Having all links from SAD to UDDs are available and aligned with each other, there are also UTS (Unit Test Specification) [15] documents consist of test plan and test procedure. Developers are responsible for adding unit test cases corresponding to the function which is tested.

Working with distributed teams in the traditional V-Model which is enhanced version of a waterfall model [16] would prevent improvements in development process. Making a change in the process would not be easy while working on the documents and implementing the related functionality. As a benefit of SAFe, our teams could interact with different levels and aspects of the system. For instance, it is discovered in a sprint retrospective meeting that a wrongly designed software component had been causing time loss during peer review process. A common unit would be better used and compiled for different hosts having different OS running on it, because that unit was not dependent on the target host. Our team took an action item and created a technical dept story to provide a solution. This story enabled the team to modify the architecture of the software component and avoided the unnecessary efforts spent for peer review process.

Table 1. Example review times after improvement

Review Id	Component	Start Date	End Date	Work Hour	Review Type	Participants	Files	Defects
3537	Persistency	13/06/18	24/10/18	120 hours	Formal Review	13	5	2
5106	Persistency	17/04/19	25/04/19	24 hours	Formal Review	12	9	21

As it is shown in Table 1, ideal review processes for the same component took less time with the similar set of participants, and this time includes both review and rework time. Although the complexity in the Review 5106 was higher than Review 3537, a considerable amount of time saved as a result of the improvement in the component.

4.2. Incompatibilities when reviewing code and document independently

The author is supposed to create a review task both for the document and the code after completing the code implementation. Ideally the document reviewer and the code reviewer should be the same person, but this is usually not the case in practice. Not having the same author in both the code review and the document review causes inconsistencies between the implementation and the documentation.

At earlier stages of the review process, code review and document review of the software units (UDD) was held independently. Once it has been addressed by our team in one of the sprint retrospective meetings, the team decided to combine the code reviews and the document reviews. This action accelerated the team velocity by saving a significant amount of time.

Table 2. Example review times for the same unit including code and UDD separately* and together*

Review Id	Component	Start Date	End Date	Work Hour	Review Type	Participants	Files	Defects
4119*	Persistency	04/10/18	22/11/18	32 hours	Formal Review (Code)	15	8	23
4095*	Persistency	01/10/18	30/10/18	32 hours	Formal Review (UDD)	14	5	17
5110**	Persistency	17/04/19	03/05/19	28 hours	Formal Review (UDD + Code)	10	9	9

Table 2 gives an example of the time spent for two different code reviews with ideal technical complexities. The total time spent for Review 4119 and Review 4095 is apparently higher than the time spent for Review 5110. The main reason behind the delay with the separate reviews is the number of defects had to be resolved back and forth in both reviews.

There is another type of incompatibility addressed by the team in sprint retrospective meetings. The root cause of this incompatibility was mainly the difference in coding rules between ASIL software components and QM software components. Since an ASIL D unit complies with the highest functional safety level functional safety awareness is required for a reviewer to complete review process for a higher degree ASIL SWC. If an engineer without enough experience on safety measurements is assigned to a review for a change in an ASIL D component, heavy delays occurs until the review is accepted and closed. As a result, another engineer with enough knowledge is mostly involved in the review.

4.3. Working with distributed teams

Working with distributed teams has always been a challenge in terms of management. The automotive industry is confronted with the management challenge of multi-tier outsourcing approaches. As a part of our business with other global organizations we also experience some impediments due to being geographically distributed. Our team identified the root causes of being a remote team in a Problem-solving workshop which is a part of Inspect and Adapt meetings. The team voted on the root causes to select the

most significant factor causing the problems in accordance with SAFe practises. In this subsection the selected root causes are described, and the selected action points are listed.

Time zone differences is one of the major reasons of the impediments. Arranging meetings can easily be a challenge due to the office hour differences. If you build up a scrum team meeting for a daily stand-up, you need to be careful to keep all team members comfortable enough to be available every day at the same time at the same place (a communication channel in this case).

Lack of face-to-face communication is another obstacle to building collaboration. The scrum team should be proactive, responsive and open to minimize isolation. Nowadays, conference and collaboration tools are quite enough to provide the environment for immediate contact between team members. However, scrum masters should spend extra effort to build such a team habit of quick calls.

Office culture can also vary in different countries and regions. Having different conventions can easily cause frustration between team members. Such as having more annual leaves, having different dress code, company benefits, flexibility in working hours and ease of access to some equipment are some example reasons for a possible frustration. As scrum encourages cross functionality, creating fair environments in different offices should be considered while forming distributed teams.

As face-to-face communication is more effective than text messages, building high collaboration between the distributed team members requires extra effort. In order to reduce the negative impact of not being co-located to other teams these actions are taken:

- Organizing workshops more frequently in different offices
- Scheduling onboarding trainings in headquarters
- Trying to form the teams co-located as much as possible
- Pairing senior engineers with new joiners
- Rotating team members between different teams

These actions improved the overall team performance especially in the onboarding process of the new engineers. Although the projects and the topics are highly variative in complexity, there is a considerable performance gain required for a new engineer to be operative on a domain. Initially the onboarding process used to take from 6 weeks to 3 months. After the improvements, the onboarding process rarely took more than 3 weeks.

5. Conclusion

In this study we presented our analysis on the use of agile and SAFe methodologies in the automotive software development. We identified some team level obstacles of being part of large-scale global organizations in automotive. Our experience with these obstacles is explained under several topics such as working with distributed teams, incompatibilities in reviews, agility in V-model. The study also explains how we coped with these problems by using V-model in combination with agile practices.

Having adopted SAFe practices in embedded automotive software development provided better agility and predictability, better organizational structures with huge teams, faster software release cycles without loss of quality. Such an environment allows engineers to have a better awareness of the complete picture in enterprise. Applying SAFe on highly complex software systems is sometimes formidable in

practice as we mentioned about the experience within third section. Although there are other agile frameworks which address this complexity, companies should adapt the agile framework in order to have best fit and value [3].

References

1. Maarit Laanti. 2014. Characteristics and principles of scaled agile. In Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation. Springer, 9–20.
2. Leffingwell, D.: Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley (2011) ISBN-10: 0-321-63584-1, ISBN-13: 978-0-321-63584-6
3. C. Ebert and M. Paasivaara, "Scaling Agile," in *IEEE Software*, vol. 34, no. 6, pp. 98-103, November/December 2017.
4. VersionOne. 2019. 13th Annual State of Agile Report. WWW page accessed 2019-06-20. (2019). https://www.stateofagile.com/#ufh-i-521251909-13th-annual-state-of-agile-report/473508
5. M. A. Razzak, I. Richardson, J. Noll, C. Nic Canna and S. Beecham, "Scaling Agile across the Global Organization: An Early Stage Industrial SAFe Self-Assessment," *2018 IEEE/ACM 13th International Conference on Global Software Engineering (ICGSE)*, Gothenburg, 2018, pp. 116-125.
6. S. M. Sundharam, S. Altmeyer, and N. Navet. Model interpretation for an AUTOSAR compliant engine control function. In Proc. 7th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS), Toulouse, France, July 2016.
7. J. Schäuffele and T. Zurawka, Automotive software engineering: principles, processes, methods, and tools. Warrendale, Pa: SAE International, 2005.
8. Chakraborty S, Ramesh S (2015) Guest editorial special section on automotive embedded systems and software. IEEE Trans Comput Aided Des Integr Circuits Syst 34(11):1701–1703
9. A. Haghighatkhah, M. Oivo, A. Banijamali and P. Kuvaja, "Improving the State of Automotive Software Engineering," in IEEE Software, vol. 34, no. 5, pp. 82-86, 2017.
10. P. Hohl et al., “Forces That Prevent Agile Adoption in the Automotive Domain,” Proc. 17th Int’l Conf. Product-Focused Software Process Improvement (PROFES 16), 2016, pp. 468–476.
11. G. K. Hanssen, "Agile software product line engineering, DT8100 Essay," 2007. http://www.idi.ntnu.no/emner/dt8100/Essay2007/hanssen-essay07.pdf
12. R. Kurmann, "Agile SPL–SCM -- Agile Software Product Line Configuration and Release Management," in APLE 2006 Workshop collocated with SPLC2006 Baltimore, MD, USA.
13. B. Boehm and R. Turner, "Using Risk to Balance Agile and Plan-Driven Methods," IEEE Software, vol. June 2003, 2003.
14. S. Thiel et al., “Software Product Lines in Automotive Systems Engineering,” SAE Int’l J. Passenger Cars—Electronic and Electrical Systems, vol. 1, no. 1, 2009, pp. 531–543.
15. P. A. Stocks and D. A. Carrington, "Test templates: a specification-based testing framework," Proceedings of 1993 15th International Conference on Software Engineering, Baltimore, MD, USA, 1993, pp. 405-414.
16. Georgios Theocharis , Marco Kuhrmann , Jürgen Münch , Philipp Diebold, Is Water-Scrum-Fall Reality? On the Use of Agile and Traditional Development Practices, Proceedings of the 16th International Conference on Product-Focused Software Process Improvement, December 02-04, 2015, Bolzano, Italy
17. Olcaysoy, Ayşe Buharalı, and Oya Kalıpsız. "Proje Yöneticilerinin Yazılım Projelerinin Başarısı Üzerindeki Etkisi, 2018.
18. Salo, Outi, and Pekka Abrahamsson. "Agile methods in European embedded software development organisations: a survey on the actual use and usefulness of Extreme Programming and Scrum." IET software 2.1 (2008): 58-64.

19. Holtmann, Jörg, Jan Meyer, and Matthias Meyer. "A seamless model-based development process for automotive systems." Software Engineering 2011–Workshopband (2011).
20. Eklund U., Holmström Olsson H., Strøm N.J. (2014) Industrial Challenges of Scaling Agile in Mass-Produced Embedded Systems. In: Dingsøyr T., Moe N.B., Tonelli R., Counsell S., Gencel C., Petersen K. (eds) Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation. XP 2014. Lecture Notes in Business Information Processing, vol 199. Springer, Cham
21. Chitnis, K., Mody, M., Swami, P., Sivaraj, R., Ghone, C., Biju, M.G., Narayanan, B., Dutt, Y., Dubey, A.: Enabling functional safety ASIL compliance for autonomous driving software systems. Electron. Imaging 19, 35–40 (2017)
22. ISO/IEC 15504 International Standard "Information Technology – Software Process Assessment: Part 1–Part 5" 2006.
23. H. Heinecke J. Bielefeld K.-P. Schnelle N. Maldener H. Fennel O. Weis T. Weber J. Ruh L. Lundh T. Sandn P. Heitkmper R. Rimkus J. Leflour A. Gilberg U. Virnich S. Voget K. Nishikawa K. Kajio T. Scharnhorst B. Kunkel "AUTOSAR—Current results and preparations for exploitation" Proc. 7th EUROFORUM Softw. Veh. 2006-May.

Çevik yöntemlerde büyüklük ölçümü: bağımsız ölçümler mümkün mü?

Tuna Hacaloglu [1,2], Aylin Deveci [3], Selami Bagriyanik [3], Onur Demirors[4]

[1] Atılım Üniversitesi, Ankara, Türkiye
tuna.hacaloglu@atilim.edu.tr
[2]Orta Doğu Teknik Üniversitesi, Ankara, Türkiye
[3]Turkcell Teknoloji, İstanbul, Türkiye
{aylin.deveci, selami.bagriyanik}@turkcell.com.tr
[4] İzmir Yüksek Teknoloji Enstitüsü, İzmir, Türkiye
onurdemirors@iyte.edu.tr

Özet. Çevik yöntemler, günümüz yazılım sektöründe sıklıkla kullanılmaktadır. Bu yöntemler, sık aralıklarla çalışan ürün ortaya çıkarmayı ve müşteri talep ve değişiklerine olabildiğince uyum göstermeyi hedefler. Bu durumun bir sonucu olarak, Çevik yöntemlerle geliştirilen yazılım projelerinde, gereksinimlerin detaylı dokümantasyonu ikinci planda kalmış olup; geliştiriciler gereksinimleri daha az detaylı yazma eğilimindedirler. Ancak gereksinimlerin bu şekilde yazılması, işlevsel büyüklük ölçümü, ve büyüklüğe bağlı sistematik efor kestirimi gibi süreçlerin uygulanabilirliğini zorlaştırmaktadır. Bu çalışmada, Scrum pratikleri uygulanarak geliştirilen büyük ölçekli bir teknoloji ve iletişim hizmetleri sağlayıcısına ait bir yazılım projesinde iki farklı ölçer tarafından yapılan işlevsel büyüklük ölçümü deneyimleri aktarılmıştır. Temel olarak, projenin iki ayrı bakış açısından ölçülen işlevsel büyüklük değerlerinin ne kadar değişkenlik gösterdiği ve bu değişikliklerin sebeplerinin irdelenmesi hedeflenmiştir. Projenin fonksiyonel gereksinimleri önce bağımsız bir ölçer tarafından COSMIC Fonksiyonel Büyüklük Ölçüm Yöntemi ile ölçülmüştür. Daha sonra, aynı gereksinimler, projenin teknik ürün yöneticisi tarafından aynı yöntem ile ölçülmüştür. Sözü edilen iki ayrı perspektiften ölçümlerin ardından, bu iki farklı ölçüm sonuçları birbiriyle karşılaştırılmış ve ölçüm değerlerinde oluşan sapmaların olası sebepleri araştırılmıştır. Bu ölçümler ışığında oluşturulan efor kestirim modelinin başarısının da çalışma kapsamında irdelenmesi hedeflenmiştir.

Anahtar Kelimeler: çevik, agile, SCRUM, işlevsel büyüklük, COSMIC

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Size measurement in Agile Methods: are independent measurements possible?

Tuna Hacaloglu [1,2], Aylin Deveci [3], Selami Bagriyanik [3], Onur Demirors[4]

[1] Atılım Üniversitesi, Ankara, Türkiye
tuna.hacaloglu@atilim.edu.tr
[2]Orta Doğu Teknik Üniversitesi, Ankara, Türkiye
[3]Turkcell Teknoloji, İstanbul, Türkiye
{aylin.deveci, selami.bagriyanik}@turkcell.com.tr
[4] İzmir Yüksek Teknoloji Enstitüsü, İzmir, Türkiye
onurdemirors@iyte.edu.tr

Abstract. Agile methods are being used frequently in today's software industry. In these methods, more care is taken to deliver working software products in short time intervals and to adapt to customer change requests as much as possible. As a result of this, detailed documentation of requirements is pushed into the background in agile and developers have tendency of documenting requirements in a less detailed format. However, this issue complicates the applicability of functional size measurement and hence size-based effort estimation. In this study, experiences of applying functional size measurement by two measurers on a software project developed using Scrum practices in a large-scale technology and communications services provider. Basically, it is aimed to examine how the functional size values measured from two different perspectives vary and possible reasons for these changes. First, functional requirements of the project are measured by an independent measurer with COSMIC Functional size measurement method. Later, the same requirements set is measured with the same measurement method by the technical product manager of the project. Following the aforesaid measurements, the different functional size values are compared and the possible causes for the variations in the measurement values are investigated. In addition, it was aimed to examine the success of effort estimation model created with software size values measured by COSMIC method.

Keywords: agile, SCRUM, functional size measurement, COSMIC

1 Giriş

Çevik yöntemler geleneksel yazılım geliştirme süreç modellerinin katı yapılarına bir alternatif olarak 2001 yılında bir manifesto [1] olarak ortaya çıkmış olup; özellikle çalışan ürünü müşteriye hızlıca sunmayı hedeflemeleri sebebiyle günümüzde sıklıkla tercih edilmektedirler. 2015 yılında Garousi, Coskuncay, Betin-Can ve Demirors tarafından yapılmış olan Türkiye'deki Yazılım Mühendisliği Pratikleri araştırmasına katılan 202 yazılım uygulayıcısından edinilen verilere göre Çevik/Yalın yöntemlerin Şelale

Modeli ’nden sonra en çok kullanılan süreç modeli olduğu ortaya çıkmıştır [2]. Çevik Yöntemler, yazılım yaşam döngüsü boyunca bir sonraki sürece ancak bir önceki süreç bütünüyle tamamlandığında geçilebilen, her sürecin kapsamlı dokümantasyonun dayatıldığı [3] ve değişiklik taleplerinin yönetilememesi gibi Şelale Modeline ait çeşitli sorunları ortadan kaldırmak üzere bir seçenek olarak benimsenmiştir. Çevik yaklaşımlarda bireyler ve onların etkileşimleri, süreçler ve araçlardan; çalışan yazılım ürünü, kapsamlı dokümantasyondan; müşteri ile işbirliği, sözleşmelerden ve değişikliklere cevap verebilmek, bir planı takip ediyor olmaktan daha değerli olarak nitelendirilmiştir [1].

Öte yandan, bir yazılım projesinin başarılı olarak değerlendirilebilmesi için, yazılımın müşteri ihtiyaçlarını tam anlamıyla yansıtmasına ek olarak, zamanında ve ön görülen bütçe dâhilinde müşteriye teslim edilmesi de gerekmektedir [4]. Diğer bir deyişle, yazılım proje geliştirme sürecinin nasıl yönetildiği, projenin başarısı açısından büyük önem taşır. Sistematik bir ölçüm sunan COSMIC İşlevsel Büyüklük Ölçümü (İBÖ) gibi yöntemler, tedarikçi ve iş ortakları ile maliyetler konusunda el sıkışmak; daha gözlenebilir, objektif ve hesap verilebilir olduğu için denetleme ve satın alma süreçleri gibi süreçleri yönetmek açısından faydalı ve gereklidir [5]. Buna ek olarak, proje yönetiminin ana kalemlerinden olan efor kestirimi, projenin geliştirme takvimini belirleme ve de maliyet hesabını doğru yapabilmek için kritiktir. Bu bağlamda, zaman, bütçe ve efor gibi kestirimleri projenin başlangıç evrelerinde gerçekleştirmek için işlevsel büyüklük ölçümünün faydalı bir girdi olduğu bilinen bir gerçektir. Buna rağmen, çevik yazılım geliştiricilerin, genellikle öznel kestirim yöntemlerini daha fazla benimsedikleri ve bunlar arasında en çok öne çıkan kestirim yöntemlerinin hikâye puanı -“story point” (HP) ve kullanım durum puanı -“use case point (KDP) olduğu gözlemlenmiştir [6]. Bu tür öznel yöntemler oldukça kabul görmesine rağmen, çeşitli araştırmacılar tarafından eleştirilmektedirler. Örneğin, Commeyne vd. [7] kestirimlerin öznel olduklarını, dolayısıyla beceri ve deneyimden bir hayli etkilendiklerini, büyüklük ölçümü olmadıkları için proje üretkenliğini belirleme konusunda eksik kaldıklarını ve projeler arası performansların karşılaştırılmasına imkân vermediklerini vurgulamıştır.

Buna karşılık, çevik yazılım projelerinde, işlevsel büyüklük ölçümünü uygulamış güncel çalışmalar mevcuttur. Son beş yılda bu konuda yapılan bazı çalışmalara baktığımızda [7,8,9,10] çevik projelerde, COSMIC işlevsel büyüklük ölçümü ile yapılan efor kestirimlerinin başarılı sonuçlar verdiği görülmektedir. Ancak, yine de öznel kestirim yöntemlerinin çevik projelerde daha çok tercih edildiği [6] dikkate alındığında işlevsel büyüklük ölçümünün efor ve benzeri kestirimleri yapmak üzere başarısının araştırılması ve yazılım endüstrisinde daha az kullanılmasının sebeplerinin değerlendirilmesi önem taşır.

Çevik projelerde işlevsel büyüklük ölçümünün kullanımına bağlı literatürde dile getirilen bir takım güçlükleri 2018 yılında, Hacaloglu ve Demirors bir literatür taraması kapsamında derlemiştir [11]. Buna ek olarak, Hacaloglu ve Demirors, 2019 yılındaki başka bir çalışma kapsamında [12], üç farklı organizasyonun, çevik yöntemler kullanılarak geliştirilen projelerinde, COSMIC işlevsel büyüklük ölçümü yöntemi ile ölçüm yaparken karşılaşılan zorlukları çoklu vaka çalışması yaparak irdelemişlerdir. Çalışma

sonucunda, literatür bulgularına paralel olarak, COSMIC işlevsel büyüklüğü için gerekli olan bazı bileşenlerin ayırt edilmesinde gereksinimlerin yazılış tarzından kaynaklı bazı güçlüklerin olduğu gözlenmiştir [12].

Bu durumun bir sebebi olarak, çevik yöntemleri benimseyen yazılım geliştiricilerin, dokümantasyona daha az; çalışan ürünün hızlı bir biçimde teslimine ise olabildiğince çok önem verme bakış açısında oldukları [3] ve bu kişilerin gereksinimleri daha az detay içerecek biçimde yazma eğiliminde olmaları gösterilebilir [13]. Sonuç olarak gereksinim tanımlarının formal bir yapıda olmayışı COSMIC gibi işlevsel büyüklük ölçümü (İBÖ) yöntemlerinin kullanımını zorlaştırdığından [14], işlevsel büyüklüğün yazılım proje yönetimine özgü zaman, bütçe, ve efor kestirimi gibi bir takım pratikler için kullanımını da kısıtlamaktadır.

Bu bildiride sunulan çalışmada, Türkiye'de faaliyet gösteren ve gerek çevik yöntemler ile yazılım geliştirme konusunda, gerekse COSMIC ile İBÖ ölçümü deneyimi olan büyük ölçekli bir teknoloji ve iletişim hizmetleri sağlayıcısı ait tek bir projeye odaklanarak, projenin işlevsel büyüklüğünün COSMIC ile iki ayrı perspektiften ölçümü hedeflenmiştir.

Söz konusu iki perspektif şu şekildedir:

- Yazılımın büyüklüğünün, COSMIC İBÖ ile bağımsız bir ölçer tarafından gereksinimler üzerinden ölçülmesi;
- Yazılımın büyüklüğünün COSMIC İBÖ ile ölçüm deneyimi olan teknik ürün yöneticisi tarafından ölçülmesi.

Çalışma temel olarak, ölçüm sonuçlarını karşılaştırmayı, ölçümler sonucunda oluşması muhtemel sapmaları ortaya koymayı ve bu sapmaların sebeplerini irdelemeyi ve çevik proje gereksinim içeriklerinin bu konuya ne kadar etki ettiklerini tartışmayı hedeflemektedir. Ayrıca, bu ölçümler ışığında efor kestirim modeli oluşturularak, COSMIC İBÖ yöntemi ile bulunan yazılım büyüklüğünün efor tahminleme konusundaki başarısı gözlenecektir.

Bu bildirinin 2. Bölümünde araştırma tasarımı açıklanıp; 3. Bölümünde araştırmanın uygulanması, organizasyon ve ilgili proje kısaca tanıtılarak, 4. Bölümünde araştırma bulguları sunulacaktır. 5. Bölümde ise sonuçlar ve gelecekte hedeflenen çalışmalar ifade edilecektir.

2 Araştırma Tasarımı

Bu çalışmanın üç tane araştırma sorusu vardır:

Soru 1: Çevik yöntemlerle geliştirilen bir projenin farklı perspektiflerden işlevsel büyüklüğü ne kadar değişkenlik gösterir?

Soru 2: Ölçümlerde oluşan sapmaların sebepleri neler olabilir?

Soru 3: Çevik projelerde COSMIC İBÖ ile efor kestirimi ne kadar başarılıdır?

Bu sorulara cevap aramak üzere, bu bildiride sunulan çalışma kapsamında altı adımdan oluşan bir yol haritası izlenmiştir:

1- Çevik yöntemlerle geliştirilen bir projenin edinilmesi,
2- İşlevsel büyüklüğün, bağımsız ölçer tarafından COSMIC İBÖ yöntemi ile gereksinimler üzerinden ölçülmesi,

3- İşlevsel büyüklüğün, projede çalışmış ve COSMIC İBÖ yöntemi ile ölçüm deneyimi olan teknik ürün yöneticisi tarafından ölçülmesi,
4- Sonuçlarının karşılaştırılması ve olası sapmaların irdelenmesi
5- Efor kestirim modelinin oluşturulması ve başarısının değerlendirilmesi

2.1 Ölçüm yöntemi seçimi

Bu çalışmada, projenin büyüklük değerlerini ölçmek üzere COSMIC İşlevsel Büyüklük ölçümü (İBÖ) yöntemi seçilmiştir. COSMIC İBÖ, ikinci nesil bir ölçüm yöntemi olup ISO standardına uyumludur [15]. Yöntemin temeli bir ilgi nesnesine ait veri grubunu hareket ettiren veri hareketlerini saymaya dayanır. Bu veri hareketleri Entry (E), Exit (X), Read (R) ve Write (W) olmak üzere dört farklı tipten oluşur.

Entry yani Girdi işlevsel kullanıcıdan sisteme veri hareketini,
Exit yani Çıktı sistemden işlevsel kullanıcıya doğru bir veri hareketini,
Read yani Okuma kalıcı bellekten sisteme olan bir veri hareketini ve
Write yani Yazma sistemden kalıcı belleğe olan bir veri hareketini temsil eder [15].

2.2 Proje seçme ölçütleri

Araştırmada kullanılacak projenin seçiminde aşağıda belirtilen ölçütler gözetilmiştir: Öncelikle çevik yöntemlerle yazılım geliştirme deneyimine sahip bir yazılım firmasının bulunması ve bu firmadan çevik yöntemlerden biri ile geliştirilmiş bir projenin temin edilmesi hedeflenmiştir.

2.3 Ölçerlerin seçimi

Projenin daha önce de bahsedildiği gibi öncelikle COSMIC İBÖ deneyimi olan bağımsız bir ölçer tarafından, daha sonrada projede çalışmış ve COSMIC İBÖ deneyimi olan teknik ürün yöneticisi tarafından ölçülmesi amaçlanmıştır.

2.4 Veri analizi yöntemi

Projenin gereksinimlerinin işlevsel büyüklüğü ile bu gereksinimleri gerçekleştirmeye harcanan eforların arasındaki ilişkiyi değerlendirmek üzere korelasyon analizi ve regresyon analizinin kullanılması hedeflenmiştir.

3 Araştırmanın uygulanması

Araştırma için, Bölüm 2.1'de verilen proje seçme ölçütlerini gözeterek elde edilen bir proje önce gereksinimler üzerinden bağımsız ölçücü tarafından COSMIC İBÖ yöntemi ile daha sonra da teknik ürün yöneticisi tarafından aynı yöntemle ölçülmüştür. Çalış-

maya konu olan projenin özellikleri ve ölçüm sürecine dair detaylar Bölüm 3.1'de sunulmuştur. Bu bölümlerde sözü geçen proje, bu bildiri metni boyunca A projesi olarak adlandırılacaktır.

3.1 A Projesinin tanıtımı

A Projesi, Türkiye'de faaliyet gösteren büyük ölçekli bir teknoloji ve iletişim hizmetleri sağlayıcısına aittir. Firma çevik yazılım geliştirme konusunda deneyimli olup aynı zamanda COSMIC İBÖ yöntemini konusunda da deneyim sahibidir. Bu bildiride irdelenmek üzere seçilen A projesi çok kullanıcılı bir uygulama yazılımıdır. Fona ihtiyaç duyan girişimler ile destekçileri birleştiren ve aynı zamanda online alışveriş yapılabilen bir pazar yeri platformudur. Platform genel çerçevede Java ve MySQL teknolojileri ile geliştirilen WEB sitesinden, Mobil RESPONSIVE WEB sitesinden, iOS ve Android gibi mobil uygulamalardan ve diğer yan teknolojilerden oluşmaktadır. Bu bildiride sadece WEB versiyonuna ait ölçüm değerleri sunulmuştur.

Platformu geliştiren teknik ekip kadrosu 1 proje yöneticisi, 1 analist, 4 geliştirici, ve 1 testçiden oluşmaktadır. Proje geliştirme takımı, yazılım geliştirme endüstrisinde de sıklıkla tercih edilen bir Çevik yöntem yaklaşımı olan Scrum yöntemi ile çalışmıştır. Scrum yönteminin çapraz-işlevli beklentisi çalışmanın yapıldığı ekipte yetkinliklerin veya uzmanlıkların her alanda eşit dağılmaması nedeniyle esnetilmiştir. Profiller "genelleşen uzman" şeklinde uzmanlaşmaktadır. İlkesel olarak zaman zaman uzmanlıklar arası yardımlaşma ve geçişkenlikler de olacak şekilde esnek bir yapı benimsenmiştir.

Projeler, genellikle üç haftalık sprintler halinde geliştirilmektedir. Proje süreci bir iş, hata ve proje takip sistemi üzerinde oluşturulan Dijital Board üzerinden yürütülmektedir. Takım elemanlarının tümü aynı firma çalışanı olmamakla birlikte, iki firma kısmen birlikte, kısmen farklı konumlarda çalışmaktadırlar. Buna rağmen, teknik ürün yöneticisi klasik Scrum pratiklerinin, uzaktan da olsa büyük ölçüde uygulandığını belirtmiştir. Örneğin, takım her gün günlük Scrum toplantılarını telefonla yapmalarının yanında, sprint planlama ve retrospektif toplantıları gibi gerekli hallerde bu toplantılar için bir araya gelmektedirler. Buna ek olarak, acil çıkan problemler veya hata düzeltmeleri için yine telefon üzerinden görüşmeler ile toplantıları gerçekleştirmektedir.

Projenin ölçümünün gerçekleştirileceği gereksinim setinde, kullanıcı hikayesi formatında işlevsel gereksinimler ve onlara karşılık gelen efor değerlerinin bulunmaktadır. Her bir kullanıcı hikâyesi, COSMIC İBÖ yöntemi ile daha önce bahsedilen iki farklı ölçer tarafından ölçülmüştür. COSMIC İBÖ ile iki fazda ölçüm sonucunda edinilen büyüklük değerleri Bölüm 4.1'de verilmiştir.

4 Araştırma Bulguları

4.1 A projesinden elde edilen CFP değerlerinin karşılaştırılması

Her iki ölçer tarafından, COSMIC İBÖ yöntemini kullanarak kullanıcı hikayeleri üzerinden yapılan ölçümün sonuçları (CFP) ve bu kullanıcı hikayelerini gerçekleştirmek üzere harcanmış olan efor değerleri Kişi-Gün olarak Tablo 1'de sunulmuştur. Bu çalışmadaki ölçüm 15 kullanıcı hikâyesi üzerinden gerçekleştirilmiştir. Bu işlemlere toplam

68 kişi-gün efor harcanmıştır. Kullanıcı hikayesi başına düşen ortalama efor 4,5 kişi-gündür.

Tablo 1. Ölçüm sonuçları

Kullanıcı Hikâyesi	CFP değeri (Bağımsız ölçer)	CFP değeri (Teknik Ürün Yöneticisi)	Ölçerler arası fark	Efor (Kişi-Gün)
1	3	3	0	1
2	9	3	6	2
3	3	2	1	1
4	3	2	1	1,5
5	9	8	1	4
6	3	2	1	4
7	14	30	16	12
8	15	20	5	26
9	8	10	2	6
10	4	4	0	3,5
11	2	2	0	0,5
12	2	2	0	1
13	6	3	3	3
14	2	1	1	0,5
15	3	1	2	2

Ölçüm sonuçlarına göre Tablo 1'den görülebileceği üzere, 1, 10, 11 ve 12 no.'lu kullanıcı hikâyeler için bağımsız ölçer ile teknik ürün yöneticisi ölçüm sonuçları birbirinin aynısıdır. Bu kullanıcı hikâyelerinin işlevsel süreçleri irdelendiğinde 1, 11 ve 12 no.'lu hikâyenin "listeleme", 10. no.lu hikâyenin ise kullanıcıya bir "bilgi gösterimi" olduğu görülmüştür.

Diğer yandan, bu çalışmada bağımsız ölçer ve teknik ürün yöneticisi ölçüm değerleri arasında bir takım farklar gözlenmiştir. Bu farkların aşağıda belirtilen sebeplerden kaynaklandığı düşünülmektedir:

- 2. kullanıcı hikâyesi "bilgi güncelleme" tipinde bir işlevsel süreçtir. Bu kullanıcı hikâyesi için ölçümlerde oluşan fark şu sebepten kaynaklanmaktadır. Bağımsız ölçer, "güncelleme" işlevsel sürecini ölçerken, "güncelleme yapmadan önce listele" gibi bir varsayım yaparak yaklaşırken, teknik ürün yöneticisi listeleme işleminin zaten daha önce yapıldığını ve söz konusu kullanıcı hikâyesinin ve ona karşılık gelen eforun sadece güncelleme işlemine ait olduğunu belirtmiştir. Dolayısıyla gereksinimlerde bu detay belirtilmediğinden bu durum için, bağımsız ölçer, güncellemeden önce listeleme gibi daha fazladan varsayım yaparak ölçme eğilimi göstermiştir.
- 3. Kullanıcı hikâyesindeki 1 birimlik farkın sebebi, buradaki fonksiyonel sürece dair bir veri hareketinin daha önceki geliştirme süreçlerinde geliştirilmiş olmasıdır. Bağımsız ölçer bu hareketi sayarken, teknik ürün yöneticisi saymamıştır.

- 4. Kullanıcı hikâyesindeki 1 birimlik farkın sebebi, bağımsız ölçerin bilgiyi, bilginin ilgi nesnesi kullanıcı hikâyesinde açıkça belirtilmediği için veri tabanından okuduğunu varsaymasıdır, fakat teknik ürün yöneticisi bu ilgi nesnesine dair bilginin daha önce okunduğunu belirtmiştir.
- 5. Kullanıcı hikâyesindeki iki ölçer arasındaki 1 birimlik farkın sebebi için bağımsız ölçerin hata/onay mesajını söz konusu sistemin göstereceğini varsayarken, teknik ürün yöneticisinin aslında bu mesajın yazılıma entegre başka bir sistemin görevi olduğunu belirtmesi gösterilebilir. Fakat kullanıcı hikâyesinin büyüklüğüne etki eden bu bilgi kullanıcı hikâyesinde ifade edilmemiştir.
- 6. Kullanıcı hikâyesindeki yine 1 birimlik farkın sebebi, bağımsız ölçerin kullanıcı hikâyesinde kastedilen süreci farklı anlamasından kaynaklanmaktadır. Asıl ifade edilmek istenen bir fonksiyonel sürecin ön koşulu başka bir fonksiyonel süreç iken, bağımsız ölçer bu sürecin sıfırdan geliştirileceği şeklinde bir yargıya varmıştır.
- 7, 8 ve 9. Kullanıcı hikâyelerinde, gereksinimler süreç adımlarına dair az detay içerdiği için ve bağımsız ölçerin proje hakkında teknik ürün yöneticisine oranla daha az bilgi sahibi olmasından kaynaklı burada olduğu gibi bazı süreçlerin detaylarını bilmemesinden ötürü ölçüm yaparken bazı adımları atlama eğiliminde olduğu görülmüştür.
- 14. Kullanıcı hikâyesi bir işlevsel sürece oranla daha alt seviye bilgi içermektedir. Bağımsız ölçer, buradaki işlemin ek bir veri hareketini daha içerdiğini düşünürken, teknik ürün yöneticisi gerçekte öyle olmadığını ifade etmiştir. Özellikle geliştirmesi daha önce tamamlanmış kullanıcı hikâyelerinde, hikâyenin tamamının yeniden geliştirilmesine ihtiyaç duyulmadan, sadece istenen değişikliğin gerçekleştirildiği durumlarda veri hareketi sayısının azaldığı gözlenmektedir.
- 15. Kullanıcı hikâyesi için ölçümlerde oluşan farkın sebebi bağımsız ölçerin, gereksinimi tek bir ilgi nesnesi için kayıt oluşturma işlevsel süreci olarak varsayarken, teknik ürün yöneticisinin, burada kast edilenin kayıt esnasında veri tabanına yazma şeklinin değişmesi ile ilgili olduğunu ifade etmesidir.

Genel bir yargı olarak, projede çalışan teknik ürün yöneticisinin, proje ile ilgili bilgisinin bağımsız ölçerden daha fazla olması ve teknik ürün yöneticisinin ölçtüğü CFP değerlerinin daha doğru olması beklenen bir durumdur. Ancak, çevik süreçlerde gözlemlenen, geliştirme ekibinin gerek toplantılar sırasında gerek geliştirme süreci içinde yaptıkları fikir alışverişlerinin dokümantasyona yansımama ihtimali, bağımsız ölçerlerin bu konuda güçlükler yaşamalarına yol açmaktadır.

Ölçüm farklarının irdelenmesi sonucunda görülmektedir ki, COSMIC İBÖ ölçümü proje ile ilgili bilgi düzeyleri birbirinden farklı iki ölçer tarafından büyük ölçüde tutarlı sonuçlar vermektedir. İki ölçerin CFP değerleri arasında çok fazla fark bulunmamaktadır. Ölçüm farklarının en önemli sebebi, gereksinim detaylarının optimal seviyede yazılmamasıdır. Özellikle, bağımsız ölçerin proje hakkında, teknik ürün yöneticisine oranla daha az bilgi sahibi olmasından kaynaklı daha fazla varsayım yaptığı ya da bazı süreçleri atladığı görülmüştür. Burada özellikle kavramlar için aynı terminolojilerin kullanıldığı, birden fazla anlam içermeyen, çelişkili olmayan

ve dolayısıyla herkes tarafından aynı şekilde anlaşılacak biçimde gereksinimler yazmak önem taşır [16]. Buna ek olarak, projede bir veri modelinin olmayışı, veri analizine dair herhangi bir bilginin bulunmayışı, projenin bütünsel yapısını, tutulacak verilerin ilgi nesnelerini algılamayı ve dolayısıyla COSMIC ölçümü için gerekli olan veri hareketlerini belirlerken varsayım yapmaya yol açmaktadır. Benzer bir durum [12]'deki çalışmada da gözlenmiştir.

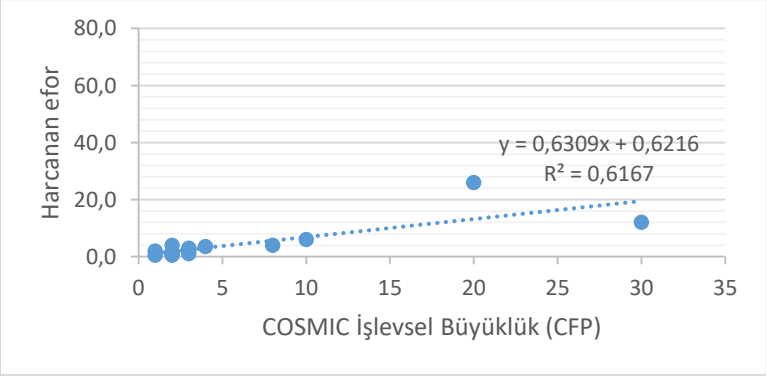
4.2 CFP değerleri ile Efor arasındaki ilişkinin analizi

Bağımsız ölçer ile teknik ürün yöneticisinin ölçümleri sonucunda yapılan görüşmelere göre karar verilen nihai ölçüm değerlerine göre kullanıcı hikayelerinin COSMIC cinsinden büyüklüğü (CFP) ile efor değerleri arasında bir ilişkinin olup olmadığı incelenmiş ve korelasyon katsayısı 0,79 olarak bulunmuştur. Bu değer güçlü pozitif ilişkiye işaret etmektedir.

Bunun ardından, basit doğrusal regresyon ile projenin işlevsel büyüklüğünün efor tahmin gücü araştırılmıştır. Doğrusal regresyon modeli (1) no.'lu denklemdeki gibi bulunmuştur.

$$Efor = 0{,}6309CFP + 0{,}6216 \qquad (1)$$

Basit doğrusal regresyon analizi sonucunda bulunan R^2 değeri 0,62 efor tahminin CFP ile yapıldığı modelde CFP değerinin efordaki % 62'lik değişimi açıklayabildiğini ifade etmektedir. CFP ile proje büyüklüğü ile efor arasındaki ilişkiyi görsel olarak ifade eden serpme diyagramı Şekil 1'de gösterilmiştir. Şekil 1'de X ekseni kullanıcı hikâyelerinin CFP cinsinden değerlerini, Y ekseni ise karşılık gelen hikayeleri gerçekleştirmek üzere harcanan efor değerlerini göstermektedir.



Şekil 1. COSMIC İşlevsel Büyüklük ve Efor dağılımı

Bu grafikte regresyon çizgisine diğer değerlere göre uzakta bulunan iki kullanıcı hikâyesi (7. ve 8. Kullanıcı hikâyelerinin) uç değer olup olmadıkları değerlendirilmiştir. 7. hikâyenin çok fazla X (Çıktı) tipinde veri hareketi içerdiği, 8. Hikâyenin ise diğer

kullanıcı hikâyelerine göre daha fazla süreç içerdiği gözlemlenmiştir. Fakat bu iki kullanıcı hikâyesini örneklemden çıkarmak üzere bir gerekçe gözlenmemiştir.

5 Sonuçlar ve Gelecekte Hedeflenen Çalışmalar

Bu bildiride sunulan çalışma kapsamında, çevik yöntemlerle büyük ölçekli bir teknoloji ve iletişim hizmetleri sağlayıcısı bünyesinde geliştirilen bir yazılım projesi iki farklı açıdan ölçülmüştür. Öncelikle, proje bağımsız bir ölçer tarafından gereksinim seti üzerinden ölçülmüştür. Ardından aynı set, proje ekibinden proje hakkında detaylı bilgi sahibi olan bir teknik ürün yöneticisi tarafından ölçülmüştür. Ölçüm sonuçları birbirleri ile kıyaslanmıştır. CFP değerlerinin büyük ölçüde benzeştiği görülmüştür. Buna rağmen, bazı CFP değerlerinde farklar mevcuttur. Ortaya çıkan bu sapmaların nedenleri araştırılmıştır. Gereksinimlerin az detay içerecek şekilde yazılmaları, eksik/ yanlış anlamalara yol açmaktadır, dolayısıyla açık ve tek bir manaya gelecek şekilde yazma yaklaşımının benimsenmesi, gereksinimler ile ilgili detayların olabildiğince dokümantasyona yansıtılmasının ölçüm tutarlılığı ve doğruluğu açısından kritik olduğu gözlemlenmiştir. COSMIC İBÖ açısından önemli olan veri analizi için eksiklerin bulunması ölçüm esnasında varsayım yapma durumuna yol açtığı görülmüştür. Daha sonra, bu ölçümler ile efor kestirim modeli oluşturulmuş ve modelin başarısı değerlendirilmiştir. CFP değerleri ile harcanan efor arasında pozitif ve güçlü bir korelasyon göz çarpmıştır. Kurulan regresyon modelinde CFP'nin eforda %62'lik bir değişimi açıkladığına dair bulgu gözlenmiştir. Bundan sonraki çalışmalarda, projenin mobil versiyonu benzer şekilde incelenecektir. Yeni bir boyut olarak bitmiş proje üzerinden ölçüm değerleri, gereksinim üzerinden ölçüm değerleri ile arasındaki değişkenlik incelenecektir.

Kaynakça

1. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Kern, J.: Manifesto for agile software development (2001).
2. Garousi, V., Coşkunçay, A., Betin-Can, A., & Demirörs, O.: A survey of software engineering practices in Turkey. *Journal of Systems and Software*, vol. *108*, 148-177. (2015).
3. McCormick, M.: Waterfall vs. Agile methodology. *MPCS, N/A*. (2012).
4. The Standish Group International, Inc. Standish Grup Chaos Report, (1995). https://www.csus.edu/indiv/r/rengstorffj/obe152-spring02/articles/standishchaos.pdf (Son erişim: 19 Haziran 2019).
5. Deveci, A., Caliskan, S. Bagriyanik, S., Gazdagi, O., Karahoca, A.: Yazılım Geliştirme Hizmetlerinin Satın Alınması için Bir Maliyet Modeli Önerisi, Ulusal Yazılım Mühendisliğ Sempozyumu (2018).
6. Usman, M., Mendes, E., Weidt, F., & Britto, R.: Effort estimation in agile software development: a systematic literature review. In Proceedings of the 10th international conference on predictive models in software Engineering, pp. 82-91, ACM. (2014).
7. Commeyne, C., Abran, A., & Djouab, R.: Effort estimation with story points and cosmic function points-an industry case study. *Software Measurement News*, vol.*21* issue 1, 25-36. (2016).

8. Ungan, E., Çizmeli, N., & Demirörs, O.: Comparison of functional size based estimation and story points, based on effort estimation effectiveness in SCRUM projects. In *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications* pp. 77-80. IEEE (2014).
9. Salmanoglu, M., Hacaloglu, T., & Demirors, O.: Effort estimation for agile software development: Comparative case studies using COSMIC functional size measurement and story points. In *Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement* (pp. 41-49). ACM (2017).
10. Ertaban, C., Gezgin, S., Bağrıyanık, S., Albey, E., & Karahoca, A.: Çevik yöntemlerde cosmic işlev puanı ve hikaye puanının birlikte kullanımı. In *CEUR Workshop Proceedings*. CEUR-WS. (2017).
11. Hacaloğlu, T., & Demirörs, O.: Challenges of using software size in agile software development: A systematic literature review. IWSM Mensura 2018, CEUR Workshop Proceedings. (2018).
12. Hacaloglu, T. and Demirors, O.: Measureability of functional size in Agile software projects: Multiple case studies with COSMIC FSM, Acccepted in Euromicro SEAA (2019).
13. Firesmith, D.: Specifying Good Requirements, in Journal of Object Technology, vol. 2, no. 4, pp. 77-87. (2003). http://www.jot.fm/issues/issue_2003_07/column7
14. Hussain, I., Kosseim, L., and Ormandjieva, O.: Approximation of COSMIC functional size to support early effort estimation in Agile. Data & Knowledge Engineering, 85, (pp.2-14). (2013).
15. COSMIC, "The COSMIC Functional Size Measurement Method Measurement Manual (The COSMIC Implementation Guide for ISO/IEC 19761: 2017) Version 4.0.2," Common Software Measurement International Consortium, (2017).
16. Firesmith, D.: Generating Complete, Unambiguous, and Verifiable Requirements from Stories, Scenarios, and Use Cases. Journal of Object Technology, 3(10), 27-40. (2004).

Heterojen Veritabanı Yönetim Sistemleri için Bir Tekil Veri Erişim Çerçevesi

Nail Diker[1], Görkem Giray[2][0000-0002-7023-9469], Murat Osman Ünalır[3][0000-0003-4531-0566]

[1] Logo Yazılım San. ve Tic. A.Ş., İzmir
[2] Bağımsız Araştırmacı, İzmir
[3] Ege Üniversitesi, Bilgisayar Mühendisliği Bölümü, İzmir
nail.diker@logo.com.tr, gorkemgiray@gmail.com,
murat.osman.unalir@ege.edu.tr

Özet. Farklı veri saklama ve sorgulama gereksinimleri için farklı saklama modelleri ve bu modelleri gerçekleyen çok sayıda araç bulunmaktadır. Bu araçlar veri saklama ve sorgulama konusundaki problemler için çözümler sunsa da diğer taraftan farklı araçlarda saklanan verilerin bütünleştirme problemi karşımıza çıkmaktadır. Bu problem için de çeşitli çözümler önerilmiştir. Bu çalışmada literatürde bulunan veri çeşitliliği ve bu veri çeşitliliği ile başa çıkmak için önerilen çözümler sınıflandırılarak incelenmiştir. Bu inceleme sonucunda yapılan çözümlemeler doğrultusunda veri çeşitliliği problemini çözmek için bir tekil veri erişim çerçevesi sunulmuştur. Bu tekil veri çerçevesinin faydası örnek bir durum senaryosunda gösterilmiştir.

Anahtar Kelimeler: Veri Yönetim Üst Modeli, Melez Veri, Melez Veri Erişim Mimarisi, Veri Çeşitliliği.

A Single Data Access Framework for Heterogeneous Database Management Systems

Nail Diker[1], Görkem Giray[2][0000-0002-7023-9469], Murat Osman Ünalır[3][0000-0003-4531-0566]

[1] Logo Software, Izmir
[2] Independent Researcher, Izmir
[3] Ege University, Department of Computer Engineering, Izmir
nail.diker@logo.com.tr, gorkemgiray@gmail.com,
murat.osman.unalir@ege.edu.tr

Abstract. There are different data persistency models for different data storage and query requirements, and there are many tools that implement these models. Although these tools provide solutions for problems in data storage and querying, on the other hand, the problem of integration of data stored in different

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

tools is encountered. Various solutions have been proposed for this problem. In this study, data variety problem and the solutions proposed to deal with this problem are classified and examined. As a result of this analysis, a single data access framework is presented to solve the problem of data variety. The benefit of this single data access framework is illustrated in an exemplary case scenario.

Keywords: Data Management Metamodel, Hybrid Data, Hybrid Data Access Architecture, Data Variety.

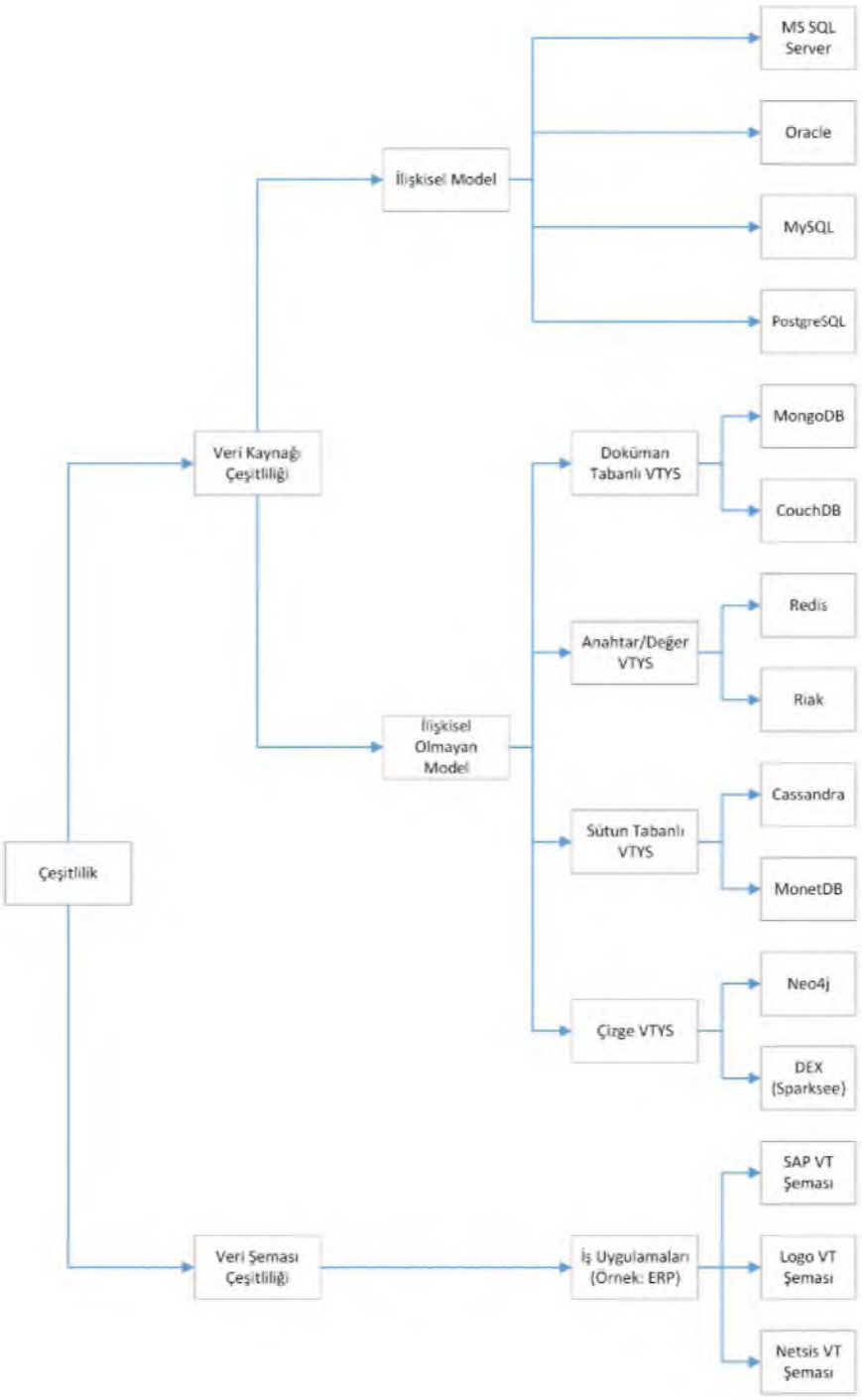
1 Giriş

Veri hacminin, çeşitliliğinin, oluşma hızının artmasına paralel olarak verinin saklanması ve sorgulanması için farklı gereksinimler ortaya çıkmıştır. Bu gereksinimlerden bazılarını ilişkisel veritabanı yönetim sistemleri (VTYS) karşılayamamıştır. Bunun üzerine farklı gereksinimler için farklı veri saklama modelleri ve bu modelleri gerçekleştiren farklı araçlar ortaya çıkmıştır. Farklı gereksinimler için farklı VTYS'ler kullanılsa da bu veri depoları arasında çeşitli ilişkiler kurulması, farklı veri depolarında saklanan verilerin bütünleştirilmesi gereksinimi birçok durumda karşımıza çıkmaktadır. Bu problem için çeşitli çözümler önerilmiştir. Bu çalışmada literatürde bulunan veri çeşitliliği ve bu veri çeşitliliği ile başa çıkmak için önerilen çözümler sınıflandırılarak incelenmiştir. Bu inceleme sonucunda yapılan çözümlemeler doğrultusunda veri çeşitliliği problemini çözmek için bir tekil veri erişim çerçevesi sunulmuştur. Bu tekil veri çerçevesinin faydası örnek bir durum senaryosunda gösterilmiştir.

2 Problem Tanımı: Veri Kaynağı ve Veri Şeması Çeşitliliği

Son yıllarda elektronik ortamda saklanan verinin öneminin ve hacminin artmasıyla birlikte veri saklama yöntemlerinin ve araçlarının çeşitliliği de bununla paralel olarak artmıştır. Veri saklama için sunulan yöntemlerdeki ve araçlardaki çeşitliliği iki ana başlık altında sınıflandırmak mümkündür: (1) Veri kaynağı çeşitliliği ve (2) Veri şeması çeşitliliği. Şekil 1 veri kaynağı ve veri şeması çeşitliliği altındaki yaklaşımları ve bu yaklaşımları benimseyen araçları göstermektedir.

Veri kaynağı çeşitliliği, uygulamaların oluşturduğu verilerin saklanması konusunda farklı yaklaşımlar ve bu yaklaşımları takip eden farklı araçlar olması nedeni ile artmaktadır. Genel olarak baktığımızda çoğu uygulama veri saklamak için VTYS'leri kullanmaktadır. Veri saklama yaklaşımlarına göre VTYS'ler iki ana grup altında toplanabilir: (1) İlişkisel VTYS ve (2) İlişkisel olmayan (NoSQL) VTYS.



Şekil 1. Veri kaynağı ve veri şeması çeşitliliği.

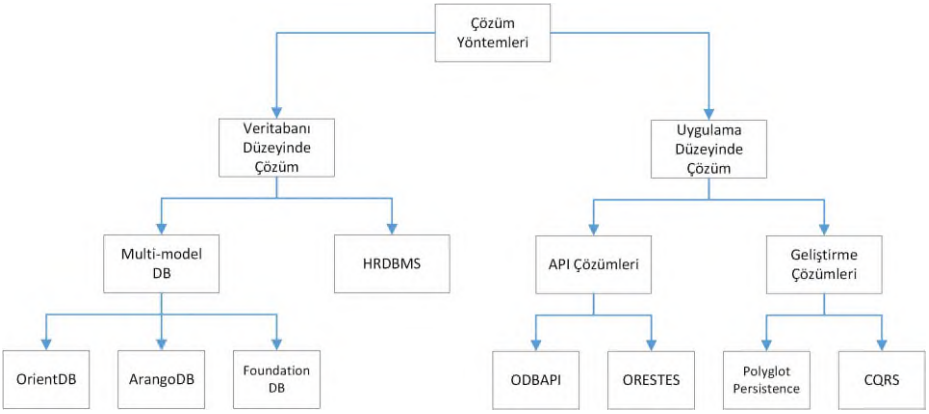
İlişkisel modeli temel alan çok fazla sayıda araç vardır. Bunlar arasında en yaygın kullanılanları Microsoft SQL Server, Oracle, MySQL ve PostgreSQL olarak sayılabilir. Bu VTYS'lerin ilişkisel modelin tablo, sütun, anahtar tanımları gibi temel kavramlarına yaklaşımları benzer olsa bile sorgulama dili özellikleri ve programlama yetenekleri farklılıklar göstermektedir. Microsoft SQL Server, T-SQL sorgulama dilini kullanırken, Oracle PL/SQL dilini kullanmaktadır.

İlişkisel VTYS'lerin ölçeklenebilirliği sınırlı olduğu ve yüksek hacimli verinin olduğu durumlarda bazı gereksinimleri karşılayamadığı için daha ölçeklenebilir olan ilişkisel VTYS'ler ortaya çıkmıştır [1]. İlişkisel olmayan VTYS'leri dört ana başlık altında incelemek mümkündür: (1) Anahtar/değer; (2) Doküman; (3) Sütun tabanlı; (4) Çizge.

Farklı modelleri benimseyen ilişkisel olmayan VTYS'ler de farklı problemleri çözmek için tasarlanmıştır [2]. Örneğin, bir anahtar ve buna karşılık gelen bir başka değerden oluşan çiftler diğer çiftlerden bağımsız olarak saklanacaksa bu durumda anahtar/değer VTYS'leri tercih edilebilir. Bu çiftlerin başka çiftlerle bağlantılı olarak saklanması gerekiyorsa bu durumda çizge VTYS'lerin tercih edilmesi daha uygun olacaktır [3].

Dört modelden birini benimseyen ve kendi içinde de farklılaşan çok sayıda VTYS bulunmaktadır. Örneğin MongoDB ve CouchDB en çok kullanılan doküman tabanlı VTYS'ler arasında yer almaktadır. [4]'e göre ilişkisel olmayan modelleri gerçekleyen araç sayısı 225'in üzerine çıkmıştır. Her 1,5 yılda bir araç sayısı iki katına çıkmaktadır [2].

Veri kaynağı ve veri şeması çeşitliliğini yönetmek için çeşitli çözümler önerilmiştir. Şekil 2'de gösterildiği gibi bu çözümleri VTYS ve uygulama düzeyinde olmak üzere iki sınıf altında incelemek mümkündür.



Şekil 2. Veri kaynağı ve veri şeması çeşitliliği için önerilen çözümlerin sınıflandırılması.

Çoklu model (multi-model) VTYS'ler tek ve bütünleşik bir arka uç üzerinden birden fazla veri modelini desteklemektedir. İlişkisel ve ilişkisel olmayan modellere göre saklanan verinin tek bir platformda yönetilmesi geliştirme, bütünleştirme, bakım ve operasyon maliyetlerini azaltmaktadır. FoundationDB, OrientDB ve ArangoDB çoklu model VTYS'ler için örnek olarak verilebilir. Çoklu model VTYS'lerin yanısıra

HRDBMS çalışması gibi melez çözümler de üretilmektedir. HRDBMS, ilişkisel VTYS'ler ile büyük veri analiz platformlarının iyi yönlerini birleştirerek kullanıma sunan yeni nesil dağıtık bir ilişkisel VTYS'dir [5].

Uygulama düzeyinde çeşitliliğe sunulan API çözümleri için ODBAPI [6] ve ORESTES [7] verilebilir. ODBAPI, bulut uygulamalarında birden fazla ilişkisel veya ilişkisel olmayan VTYS'nin kullanıldığı senaryolarda ortaya çıkan sorunları çözmek amacıyla mimari önermektedir. Mimari de RESTful servisler üzerinden standart bir uygulama geliştirme arayüzü sağlanması ve servise ulaşan isteklerin uygulama kapsamında kullanılan ilgili veritabanı yönetim sistemine ulaştırılması tasarlanmıştır [6]. ODBAPI çalışması, çoklu dilli (polyglot) yapıdaki sistemler için bir geliştirme barındırmamaktadır. Bulut tabanlı olarak bir erişim arayüzü de sunmamaktadır. ORESTES, ODBAPI'den farklı olarak Service Level Agreement (SLA) yönetimi de yapılabilmektedir [7]. Bu çalışmada, uygulama geliştirme için yaygın olarak kullanılan ORM (Object Relation Mapping) kütüphanelerinin birer servis olarak sunulması hedeflenmiştir. DynamoDB, Azure Tables gibi yeni nesil bulut tabanlı VTYS'leri kapsamaktadır.

Geliştirme çözümleri başlığı altında bulunan çoklu dilli kalıcılık (polyglot persistence) kavramı, uygulama düzeyinde farklı VTYS'lerin farklı yeteneklerini bir arada kullanmayı hedeflemektedir [8]. Yaklaşım olarak çoklu dilli kalıcılık benimsendiğinde veri tutarlılığı ve bütünlüğü hususlarına dikkat edilmelidir. İlişkisel VTYS'lerde bulunan ACID (Atomicity, Consistency, Isolation, Durability) prensibine uygun "transaction" yönetimi yerine BASE (Basic Availability, Soft-State, Eventual Consistent) prensibi çoklu dilli kalıcılık yaklaşımında benimsenmektedir. CQRS (Command Query Responsibility Segregation) tasarım deseni veri ekleme, güncelleme ve silme işlemlerini bir veri deposunda veri okuma işlemlerini de bir başka veri deposunda yapmayı önermektedir. Bu iki veri deposu arasındaki bağlantıdan sorumlu bir senkronizasyon bileşeni bulunmaktadır ve veri tutarlığını bu bileşen sağlamaktadır.

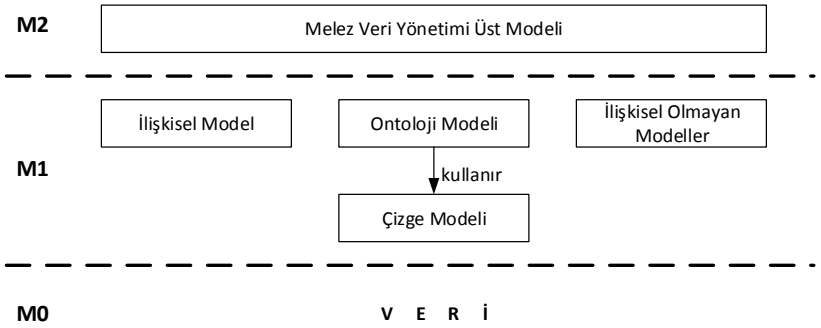
3 Çözüm Önerisi: Melez Veri Erişim Mimarisi

3.1 Melez Veri Yönetimi Üst Modeli

Tüm VTYS'lerde veri saklama yaklaşımları farklılık gösterse de soyut ve genel bir bakış açısıyla bakıldığında bazı ortak noktalar bulunmaktadır. Bu çalışma kapsamında tasarlanan melez veri yönetimi üst modelinin, modelleme hiyerarşisindeki yeri Şekil 3'te gösterilmektedir. M0 seviyesinde saklanan veri, M1 seviyesinde ise bu verinin saklandığı modele (ilişkisel, ilişkisel olmayan gibi) göre değişen üst verisi bulunmaktadır. M2 seviyesinde ise model farklılığından bağımsız olarak tasarlanmış olan melez veri yönetimi üst modeli bulunmaktadır.

Melez veri yönetimi üst modelinde ilişkisel ve ilişkisel olmayan modeller tek bir üst model altında toplanmıştır. Doküman, sütun ve anahtar/değer tabanlı veritabanları ilişkisel olmayan modeller kapsamında yer almaktadır. Çizge tabanlı veritabanları ise ayrı ele alınmıştır. Bunun nedeni çizge tabanlı dışındaki ilişkisel olmayan modellerin

aslında anahtar/değer sistemlerinin temellerini takip ederek geliştirilmiş olmasıdır. Melez veri yönetimi üst modeli kullanılarak farklı modelleri kullanan VTYS'lerin bir üst seviyede yönetilmesi hedeflenmiştir. Böylece M2 seviyesinde tüm veri saklama araçları için birleştirilmiş bir arayüz sunulması hedeflenmiştir.



Şekil 3. Melez veri yönetimi üst modelinin modelleme hiyerarşisindeki yeri [9].

3.2 Melez Veri Yönetimi Üst Modelinin Tasarımı

Melez veri yönetimi üst modelinin esnek ve genişleyebilir bir yapıda olması gerekmektedir. Değişen gereksinimler doğrultusunda, bundan önce de olduğu gibi, gelecekte farklı veri saklama yaklaşımları geliştirilecek ve bu yaklaşımların yaygınlaşmasına paralel olarak melez veri yönetimi üst modelinin bu yaklaşımları da kapsayacak şekilde genişletilmesi söz konusu olacaktır.

Melez veri yönetim üst modeli için Tozer'in geliştirdiği veri yönetimi üst modeli [10] temel alınmıştır. Tozer'in üst modeli sadece ilişkisel ve nesne tabanlı modelleri kapsamaktadır. Bu çalışma kapsamında bu üst model ilişkisel olmayan modelleri de destekleyecek şekilde genişletilmiştir.

Melez veri yönetim üst modelinin tasarım sürecinde öncelikle farklı veri saklama yaklaşımları temel alınarak tasarlanmış VTYS'lerin üst veri yapıları oluşturulmuştur. Bu kapsamda ilişkisel, anahtar/değer, doküman, sütun ve çizge tabanlı VTYS'lerinin ortak özelliklerini temsil edecek Datastore kavramı tanımlanmıştır.

Datastore temel kavramının her bir VTYS için özelleşmiş hallerini temsil edecek kavramların oluşturulmasıyla tasarıma devam edilmiştir. Örneğin, en yaygın olarak kullanılan ilişkisel VTYS'lerinde ortak olarak kullanılan kavramlar belirlenip melez veri yönetim üst modeli zenginleştirilmiştir: Veritabanı, Tablo, Satır, Sütun, Birincil Anahtar, Yabancı Anahtar.

Kullanımı giderek yaygınlaşan doküman tabanlı VTYS'leri için de aynı şekilde ortak kavramlar belirlenmiştir: Veritabanı, Koleksiyon, Doküman, Alan, Değer.

Her bir VYTS'yi temel alarak tasarlanmış kavramlar belirli bir özellik kümesine kadar ortaklaşmayı sağlamaktadır. Fakat bu yaklaşımlar kullanılarak geliştirilmiş araçlar temelinde değişen özelliklerin de melez veri yönetim üst modelinde kapsanması gerekmektedir. Örneğin, ilişkisel model kullanılarak geliştirilmiş olan Microsoft SQL Server, veritabanı işlemlerini yönetmek için "Transact SQL" (T-SQL) sorgu dilini kullanmaktadır. T-SQL, SQL standardının üzerine birtakım gelişmiş

özelliklere sahiptir. Oracle ise "Procedural Language/SQL" (PL/SQL) kullanarak ilişkisel modelin özelliklerini desteklemeyi tercih etmiştir.

Doküman tabanlı VTYS'ler için ise MongoDB ve CouchDB incelendiğinde her ikisinde de dokümanlar alanlardan oluşmakta ve her bir alanın kendine özgü değerleri bulunmaktadır. MongoDB'de dokümanlar, koleksiyonlar içerisinde saklanmaktadır. Koleksiyonlar ise veritabanları kapsamında yer almaktadır. Oysaki CouchDB'de bu tarz bir yapı bulunmamaktadır ve dokümanlar doğrudan veritabanı seviyesinde saklanmaktadır (Bkz. Tablo 1).

Tablo 1. Melez veri yönetim üst modelindeki kavramlar

İlişkisel VTYS		**Doküman Tabanlı VTYS**		**Anahtar/Değer VTYS**		**Çizge VTYS**		**Datastore**
MS SQL Server	Oracle	MongoDB	Couch DB	Redis	Riak	Neo4j	DEX (Sparksee)	Datastore individual
Database	Database	Database	Database	Instance	Instance	-	-	Data Model
Table	Table	Collection	-	Database	Database	Label	Node Type	Entity
Row	Row	Document	Document	Key/Value Pair	Key/Value Pair	Node	Node	Entity Occurrence
Column	Column	Field	Field	Key	Key	Property Key	Attribute	Attribute
Column Value	Column Value	Field Value	Field Value	Value	Value	Property Value	-	Attribute Value
Column Type	Column Type	Field Type	Field Type	Value Type	-	-	-	Attribute Type
Foreign Key	Foreign Key	-	-	-	-	Relationship	Edge Type	Relationship

Anahtar değer ikililerini barındıran ve ilişkisel olmayan VTYS'nin en temel halini temsil eden sistemlerde de ilişkisel ve doküman tabanlı VTYS'lere benzer ürünlere özelleşmiş davranışlar ve yetenekler mevcuttur. Örneğin Riak değer olarak ne türde bir veri saklandığı ile ilgili hiçbir bilgi barındırmazken Redis daha yapısal bir yaklaşım benimseyerek değer türü olarak belirli tiplere izin vermektedir.

Ürünler arasında veri yönetimi açısından işlevsel olmayan farklılıklar da mevcuttur. Riak ve Redis ürünleri bu anlamda incelendiğinde Riak'ın Microsoft Windows işletim sistemleri üzerinde çalışamadığı, Redis'in ise bu işletim sistemini de desteklediği görülmektedir. Bu anlamda doküman VTYS'leri için ise CouchDB'nin CRUD işlemleri için REST ("REpresentational State Transfer") uyumlu olarak hazır bir hizmet sunarken, MongoDB'nin böyle bir hazır hizmeti bulunmamaktadır.

Tasarım sürecinin sonucunda, Tozer'in veri yönetim üst modeli, tüm bu örneklerden hareketle ürünlere özgü bu farklılıkları kapsayacak şekilde genişletilmiştir. Elde edilen melez veri yönetim üst modeli Tablo 1'de gösterilmektedir.

Tablo 1'de "Datastore" başlıklı kolonda yer alan kavramlar Tozer'in üst modelinde bulunan kavramlardır. Bunların ilişkisel ve ilişkisel olmayan VTYS'de araç düzeyindeki kavramsal karşılıkları tabloda belirtilmiştir. Tabloda yer alan kavramlar dışında üst model Datastore kavramı ile zenginleştirilmiştir. Datastore kavramını özelleştiren kavramlar ise veri saklama yaklaşımlarına göre tasarlanmıştır.

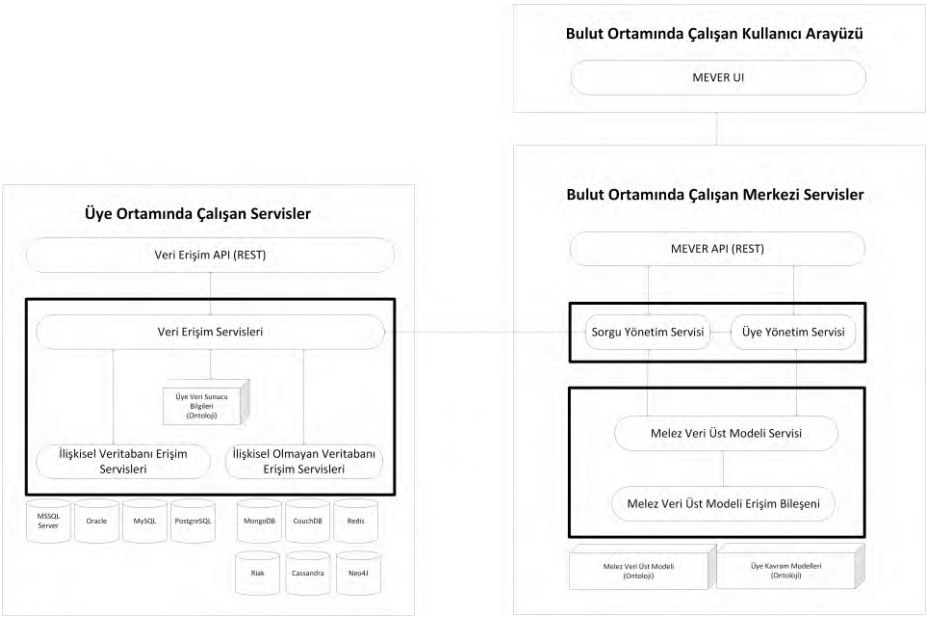
3.3 Bir Melez Veri Erişim Mimarisi Önerisi

Bir önceki bölümde açıklanan melez veri yönetim üst modelini gerçekleyen bir mimari Şekil 4'te gösterilmektedir.

Mimaride önerilen bileşenler aşağıda kısaca açıklanmıştır:

- *MEVER (Melez Veri Erişim Çerçevesi) API:* Sistemin dışa açılan yüzünü temsil etmektedir. Sistem işlevlerinin tümü bu API üzerinden sunulmaktadır. Bu işlevler, (1) Yeni üyelik işlemi; (2) Üye VTYS'lerinin tanımlanması; (3) Üye kavramlarının tanımlanması; (4) Kavram sorgulama; (5) Önerilerin alınması olarak sıralanabilir.
- *MEVER (Melez Veri Erişim Çerçevesi) UI:* MEVER API kapsamında yer alan işlevlerin kullanıcılar tarafından rahat şekilde kullanılmasını sağlayan arayüzleri barındırmaktadır.
- *Sorgu Yönetim Servisi:* Gelen sorguların iletildiği servistir. Melez veri modeli servisi ile haberleşerek üyeye özel kavram tanımı, özellikleri ve özelliklere karşılık gelen veri modeli düzeyindeki bilgileri alır. Sorguyu kavramsal düzeyden veri erişim servislerinde çalışacak VTYS sorgularına dönüştürmekle görevlidir. Bu görevinin yanı sıra veri erişim servislerinden dönen sonuç kümelerine göre sorgu cevabını oluşturacak kavram ya da kavramları oluşturur. Oluşturduğu kavram ya da kavramları MEVER API'ye iletir.
- *Üye Yönetim Servisi:* Yeni üye kayıt işlemleri, üyeye ait VTYS ve kavram tanımlarının alınması ve bu bilgilere erişilmesinden sorumludur. Bir kavram sorgusu yapıldığında üyenin bilgilerini kontrol eder.

- *Veri Erişim Servisleri:* Üyeye ait VTYS'ler ile doğrudan veya VTYS servisleri üzerinden kendisine iletilen sorguları çalıştırır ve sonuç kümelerini geri döndürür. Üyelerin kendi ortamlarında çalışacaktır. İlişkisel ve ilişkisel olmayan VTYS'lerine özelleşmiş servisler ile haberleşmektedir.
- *Melez Veri Üst Modeli Servisi:* Melez veri modeline, üye kavram ve veri sunucusu bilgilerine erişimi yöneten servistir.
- *Üye Kavram Modelleri:* Üyelere ait kavramları Melez Veri Modeline uygun şekilde saklayan ontolojilerdir.
- *Üye Veri Sunucu Bilgileri:* Üyelere ait VTYS sunucuların erişim yöntemlerini Melez Veri Modeline uygun şekilde saklayan ontolojilerdir.



Şekil 4. Melez veri erişim mimarisi.

4 Durum Çalışması

4.1 Problem

Birçok alanda çok çeşitli verinin farklı gereksinimleri yerine getirecek şekilde saklanması ve bu verinin gerektiğinde bütünleştirilerek istemcilere sunulması gerekmektedir. Böyle bir duruma örnek olarak insan kaynakları alanından bir problem seçilmiştir.

Çoğu firma, faaliyetini sürdürebilmek için çalışanlar istihdam etmektedir. Firmanın, çalışanlarıyla ilgili sakladığı verinin kapsamı firmaların faaliyet gösterdiği sektörlere ve işleyişlerine göre değişiklik göstermektedir. Genel olarak değişiklik göstermeyen veri olarak çalışanın adı, soyadı, doğum tarihi, kan grubu, vatandaşlık

numarası, fotoğrafı gibi kişiye özgü bilgiler; firmayla olan çalışma ilişkisini temsil eden veri için de sicil numarası, firmadaki pozisyonu, işe giriş tarihi, sözleşmesi ele alınabilir.

Bordrolama süreçleri için "Logo BordroPlus" ve "Netsis 3 Bordro" uygulamalarını farklı kurumlarında aynı anda kullanan holding veya şirketler grubu örnekleri ile karşılaşılmaktadır. Her iki uygulama benzer iş adımlarını gerçekleştirmek üzere kurgulandıkları için aynı kavramları farklı veritabanı desenlerinde saklamaktadır. Örneğin çalışan kavramı ile ilgili verileri, "Logo Bordro Plus" uygulaması Tablo 2'de gösterildiği gibi bir tabloda saklamaktadır. "Netsis 3 Bordro" uygulaması ise çalışan kavramıyla ilgili verileri iki ayrı tabloda saklamaktadır. "Logo Bordro Plus" uygulamasının çalışan ile ilgili Tablo 2'de gösterilen dört alandaki veriye "Netsis 3 Bordro" uygulamasında erişmek için bir görünüm (view) oluşturulmuştur.

Tablo 2. İki farklı uygulamada "çalışan" kavramıyla ilgili aynı verileri temsil eden iki farklı veritabanı nesnesi.

Logo Bordro Plus		Netsis 3 Bordro	
Tablo adı: LH_{KurumNo}_PERSON		Görünüm adı: SICIL	
Kolonlar	LREF	Kolonlar	OZLUKID
	CODE		SICILNO
	NAME		ADI
	SURNAME		SOYADI

Örnekte bahsedilen her iki uygulama da birden fazla ilişkisel VTYS'yi desteklemektedir. Örneğin bir kurum Oracle kullanırken diğer kurum MS SQL Server'ı tercih edebilir. Dolayısıyla yukarıda anlatılan veri şemasındaki çeşitliliğin yanında veri kaynağı çeşitliliği de örnek durum çalışması için söz konusudur.

4.2 MEVER Kullanılarak Oluşturulan Çözüm Önerisi

Şirketler grubu veya holding açısından bakıldığında çalışanların tekil bir arayüz üzerinden sorgulanabilmesi gerekmektedir. Bu sorunun çözümü için melez veri yönetim üst modeli kullanılarak şirketler grubuna özel bir çalışan kavramının tanımlaması Tablo 3'te gösterildiği gibi yapılabilir.

Tablo 3. Çalışan kavramına ait özellikler.

EMPLOYEE
EMPLOYEE_SYSTEM_ID
EMPLOYEE_CODE
EMPLOYEE_NAME
EMPLOYEEE_SURNAME

Bu kavramı oluşturan özellikler ile veritabanı deseninde yer alan alanlar eşleştirilerek üst model ile veritabanı yönetim sistemlerinde bulunan tabloların eşleşmesi sağlanmaktadır. Bu noktada tek bir kavramın birden fazla sistemde yer alan veri yapısı ile bütünleşik çalışabilmesi sağlanmış olmaktadır. Bordrolama uygulamalarını kullanan şirketlerin bu uygulamaları kullanım biçimlerinde herhangi bir değişiklik yapmasına gerek yoktur. Aynı zamanda teknik düzeyde bir dönüşüm, yenileme ve/veya uygulamaların yeniden geliştirilmesi ihtiyacı bulunmamaktadır.

Tekil bir arayüz üzerinden sorgulama/raporlama yapabilmek için veri senkronizasyonu veya transferi gerekmemektedir. Bununla birlikte "on-premise" olarak kendi sistemleri içinde yer alan verilere bulut ortamından, dolayısıyla İnternet erişiminin olduğu herhangi bir yerden erişim olanağı da yaratılmaktadır. Bir önceki bölümde bahsedilen veri erişim servislerinin, üyelerin kendi ortamlarında çalışacağı tasarlandığı için veritabanı sorguları yerel ağlar üzerinde çalıştırılacak ve üyelerin sadece görüntülemek istediği verilerin İnternet üzerinden transferi gerçekleştirilecektir. Veritabanı yönetim sistemlerinin dış ağlara açılma ihtiyacı bulunmadığı için güvenlik anlamında da bir dezavantaj oluşması engellenmektedir.

Eğer mevcut yaklaşım ve yöntemlerle üzerinde durulan örnek durum çalışması gerçekleştirilmek istenseydi, bir raporlama veya iş analitiği uygulaması üzerinden veritabanı yönetim sistemlerinden verilerin bu uygulamaya belirli aralıklarla aktarılması ve iş analitiği uygulaması üzerinden kavramsal olarak değil sadece bu rapora özel veri yapısının tasarlanması söz konusu olacaktır.

MEVER kullanılarak anahtar kelime tabanlı olarak kavram ve/veya bulunmak istenen anahtar kelime girilerek sorgu yapılabilir. Bu sorgu sonucunda kavramın her bir veritabanı sisteminde eşlenmiş olan veritabanı yapılarına (ilişkisel model ise tablo veya görünüm) karşılık geldiği ontoloji üzerinden tespit edilir. Bu veritabanı yapılarının içerik yapılarına uygun sorgulama cümleleri MEVER tarafından oluşturulur. Örneğin, MEVER'deki "çalışan" kavramı kullanılarak ve "Ufuk" anahtar kelimesini "sicil kodu", "adı" ve "soyadı" alanlarında arayan bir sorgu MEVER tarafından iki uygulama için aşağıdaki gibi iki sorguya otomatik olarak dönüştürülmektedir.

"Logo Bordro Plus" veritabanı için: `SELECT * FROM LH_001_PERSON WHERE CODE LIKE '%UFUK%' OR NAME LIKE '%UFUK%' OR SURNAME LIKE '%UFUK%'`

"Netsis 3 Bordro" veritabanı için: `SELECT * FROM SICIL WITH(NOLOCK) WHERE SICILNO LIKE '%UFUK%' OR ADI LIKE '%UFUK%' OR SOYADI LIKE '%UFUK%'`

Bu sorgular ontoloji tarafından hangi ilişkisel veritabanı yönetim sistemi kullanılmış ise onun desteklediği SQL kurallarına uygun olarak oluşturulur. Örneğin Oracle ise PL/SQL, MSSQL Server ise T-SQL kurallarına uygun oluşturulur. Böylelikle veri kaynağı çeşitliliği MEVER seviyesinde ortadan kaldırılmış ve sorgu detaylarının sorumluluğu MEVER'e teslim edilmiştir. Veri şeması çeşitliliği anlamında da farklı uygulamaların farklı veritabanı desenlerine uygun olarak tanımlamalar ontoloji seviyesinde bir kez yapıldığı ve oluşturulan sorgular MEVER'in dışa açılan kullanıcı arayüzü ve servis seviyelerinde bu uygulamalardan bağımsız olduğu için tekil bir görünüme ulaşılmaktadır. Yukarıda oluşturulan sorgular

ayrı ayrı uygulama veritabanlarında eş zamanlı olarak çalıştırılıp sonuçlar MEVER'in üst veri yönetim modeline uygun formata dönüştürülüp kullanıcıya tekilleştirilmiş bir şekilde sunulmaktadır.

5 Sonuçlar

Bu çalışmada veri çeşitliliği problemine çözüm olarak bir melez veri erişim mimarisi önerilmiştir. Günümüzde verinin dağıtık doğası düşünüldüğünde bu mimari servis tabanlı olarak tasarlanmıştır. Veri yönetim üst modeli farklı veri saklama yöntemlerini birleştiren soyut ve genel model olarak M2 seviyesinde konumlandırılmıştır. Mimaride bu üst model ontolojilerle gerçeklenmiştir. Melez veri erişim mimarisi geliştirilerek bu çalışmada kısaca anlatılan bir örnek durum senaryosuyla faydası gösterilmiştir.

Kaynakça

1. Martin Fowler, The Future is : Polyglot Persistence, https://martinfowler.com/articles/nosql-intro-original.pdf, 2012
2. Marcus, A. (2011). The nosql ecosystem. The Architecture of Open Source Applications, 185-205.
3. Developer Guides, https://neo4j.com/developer/graph-db-vs-nosql/#_relate_key_value_stores_with_graph_databases
4. List of NoSQL Databases, http://nosql-database.org/
5. Arnold, J., Glavic, B., & Raicu, I. (2019). HRDBMS: Combining the best of modern and traditional relational databases. arXiv preprint arXiv:1901.08666.
6. Sellami, R., Bhiri, S., & Defude, B. (2014, June). ODBAPI: a unified REST API for relational and NoSQL data stores. In 2014 IEEE International Congress on Big Data (pp. 653-660). IEEE.
7. Gessert, F., Friedrich, S., Wingerath, W., Schaarschmidt, M., & Ritter, N. (2014). Towards a Scalable and Unified REST API for Cloud Data Stores. In GI-Jahrestagung (pp. 723-734).
8. Schaarschmidt, M., Gessert, F., & Ritter, N. (2015). Towards automated polyglot persistence. Datenbanksysteme für Business, Technologie und Web (BTW 2015).
9. Diker, N., Giray, G., & Ünalır, M. O. (2016). Servis Tabanlı Bir Melez Veri Erişim Mimarisi Önerisi. UYMS.
10. Tozer, G. (1999). Metadata management for information control and business success. Artech House, Inc.

Çoklu Veri Saklama Altyapısı

Cahit Yiğit Çetiner, Merve Kaymak, Mustafa Karabacak ve Tuba Yağlı

HAVELSAN A.Ş.
Komuta Kontrol Savunma Teknolojileri
Ankara, Türkiye
{ccetiner,mkaymak,mkarabacak,tkizik}@havelsan.com.tr

Özet. Komuta Kontrol Bilgi Sistemlerinin (KKBS) öncelikli kullanım alanlarından biri cari harekâtın takibidir. Cari harekâtın sağlıklı bir şekilde takip edilebilmesi için KKBS'lerin radar, sensör gibi donanımlar ile bütünleşik çalışabilmeleri gerekmektedir. Entegrasyon sırasında alınan duraksız verilerin (streaming data) KKBS bünyesindeki ilişkisel veri tabanında saklanması, gelen verinin yoğunluğundan dolayı performans kaybına ve sergilemede gecikmelere sebep olabilmektedir. Duraksız verilerin saklanması için sütun temelli veri tabanlarının kullanımı daha etkin olsa da, mevcut uygulamanın kullandığı veri yapısının bütünüyle değiştirilmesi kabul edilemez. Çelişen isterlerin olduğu bu gibi durumlarda, çoklu veri saklama altyapıları kullanılarak istere uygun veri saklama yöntemlerinin aynı anda kullanılabileceği değerlendirilmektedir. Bu doğrultuda, HAVELSAN tarafından KKBS'leri veri saklama yönteminden soyutlayacak çoklu bir veri saklama altyapısı geliştirilmektedir. Bu makalede, hem ilişkisel veri tabanı hem de geliştirilen çoklu veri saklama altyapısı kullanılarak örnek olay çalışması üzerinden performans karşılaştırması anlatılmakta ve altyapının kullanılabilirliği tartışılmaktadır.

Anahtar Kelimeler: Komuta kontrol uygulamaları, Çoklu veri saklama altyapısı, Duraksız veri, İlişkisel veri tabanı, Sütun temelli veri tabanı

Polyglot Persistence Infrastructure

Cahit Yiğit Çetiner, Merve Kaymak, Mustafa Karabacak ve Tuba Yağlı

HAVELSAN
Command & Control and Combat Systems
Ankara, Turkey
{ccetiner,mkaymak,mkarabacak,tkizik}@havelsan.com.tr

Abstract. One of the priority usage areas of the Command and Control Information Systems (CCIS) is monitoring of the current operations. In order to monitor the current operation in a reliable way, CCIS should be able to work integrated with equipment such as radar and sensor. Storing streaming data in the relational database within the CCIS may cause loss of performance and delay in display due to the density of incoming data. Although the use of column-based

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

databases for storing nonstop data is more efficient, it is unacceptable to completely alter the data structure used by the current application. In such cases where there are conflicting requirements, it is considered that multiple data storage infrastructures can be used at the same time using appropriate data storage methods. Accordingly, HAVELSAN is developing a multiple data storage infrastructure that will isolate the CCIS from the data storage method. In this article, the performance comparison is discussed and the usability of the infrastructure is discussed by using both relational database and multiple data storage infrastructure developed.

Keywords: Command and control applications, Polyglot persistence, Streaming data, Relational database, Column-oriented database

1 Giriş

Komuta Kontrol Bilgi Sistemlerinin en önemli kullanım alanlarından biri muharebe sahasındaki muharebe elemanlarının güncel durum ve konum verileri ile istihbarat bilgilerinin takip edilmesidir. Bu kapsamda gerekli verilerin elde edilebilmesi için, dış sistemler ve donanımlar ile bütünleşik çalışabilen bir KKBS kurgusu kaçınılmazdır. Özellikle donanımların içinde bulunduğu entegrasyonlarda, KKBS'lerin gelen duraksız verileri işlemesi, saklaması ve gerçeğe yakın zamanlı sergilemesi gerekmektedir.

Çoklu Veri Saklama yöntemi, mevcut çalışan uygulamanın kullandığı veri yapısını değiştirmeden, mevcut veri saklama çözümünün yanı sıra çelişen veya darboğaz yaratan isterleri karşılamak için diğer veri saklama yöntemlerinin de çözüme dâhil edilebilmesini amaçlamaktadır. Bu sayede tek bir veri modeli üzerinden birden fazla veri saklama yöntemi sunulmaktadır.

Bu makalede çoklu veri saklama altyapısı kullanılarak örnek olay çalışması üzerinden performans karşılaştırması anlatılmakta, ayrıca altyapının geliştirilmesine dair kazanılmış deneyimler ve karşılaşılan sorunlar paylaşılmaktadır. İkinci bölümde problem tanımı verilerek, güncel durumdaki KKBS sisteminin yapısı ve karşılaşılan sorunlar anlatılmaktadır. Üçüncü bölümde çoklu veri saklama altyapı mimarisi anlatılmaktadır. Dördüncü bölümde karşılaştırmalı bir örnek olay çalışması anlatılmaktadır. Beşinci bölümde ise örnek olay çalışmasında toplanan sonuçlar ışığında sunulan çoklu veri saklama altyapısının geçerliliği tartışılmaktadır.

2 Problem Tanımı

KKBS sistemleri, hassas veriler ile çalıştıkları için bütünlük, tutarlılık, bağımsızlık ve dayanıklılık özelliklerinden dolayı genellikle İlişkisel Veri Tabanı Yönetimi Sistemi (Relational Database Management System - RDBMS) temelli çözümleri benimsemektedir. Bu makalede konu edilen örnek olay çalışmasında da, HAVELSAN bünyesinde geliştirilen, ilişkisel veri tabanı üzerinde çalışan, NATO destekli JC3IEDM (The Joint C3 Information Exhange Data Model) [6] veri modeli kullanan ve duraksız

veri üzerinden entegrasyon ihtiyacı bulunan bir KKBS'nin, duraksız verileri saklama konusunda yaşadığı problemler temel alınmış ve bu problemlere olası çözüm yolları önerilmiştir.

KKBS uygulamasının çalışması sırasında yazma ve sorgulama performansını etkileyecek en önemli etkenlerden biri kullanılan veri modelidir. KKBS uygulama altlığı olarak seçilen JC3IEDM veri modeli, asgari veri yedekliliği temel alınarak 3NF (Third Normal Form) olarak tasarlanmıştır. JC3IEDM, modelleme konusunda başarılı olsa da KKBS altlığı olarak kullanıldığı ve yoğun sorgulara maruz kaldığı durumlarda normalize yapısından kaynaklanan birleştirme (join) işlemleri nedeni ile istenilen performansı gösterememektedir.

Sorgu performansının artırılması için Şekil 1'de görüldüğü gibi gerçekleşmiş görünümler (materialized view) kullanılarak denormalize bir yapı elde edilmiş ve sorgu esnasındaki birleştirme maliyetlerinden kurtulma yoluna gidilmiştir. Bu durum, sorgu ağırlıklı ve nispeten aralıklı veri kaydı yapıldığı durumlarda kabul edilebilir bir performans sergilese de veri girişinin yoğun olduğu durumlarda yetersiz kalmaktadır. Özellikle dış veri kaynaklarından alınan duraksız verinin geçmişe yönelik saklanması veri tabanına yapılan kayıt işlemlerinin sayısını artırdığından güncel veriler üzerinden işlem yapılabilmesi için kullanılan gerçekleşmiş görünümlerin sıkça güncellenmesi gerekmektedir. Bu durum, genel sorgu performansını olumsuz etkileyerek durum sergilemede gecikmelere sebebiyet vermektedir.



Şekil 1. Güncel KKS Mimarisi

Karşılaşılan diğer bir sorun ise teknolojik olarak ödünleşen (trade-off) isterlerdir. KKBS bünyesinde işlenen duraksız verilerin geçmişe yönelik saklama isteri olduğu durumlarda (fiziksel silme, güncelleme yapılmadan) saklanan veri boyutu çok büyümekte ve verilerin işlendiği sorgular uzun süreler alabilmektedir. Bu durumda, geçmişe yönelik duraksız veri saklama ve performans isterleri arasında ödünleşme durumu ortaya çıkarmaktadır.

Karşılaşılan bu tip sorunların çözümü için veri saklama yönteminin değiştirilmesi gerekmektedir. Ancak bu yöntem hem maliyetlidir hem de kullanımda olan KKBS'ler için uygulanabilir değildir.

3 Çoklu Veri Saklama Mimarisi

Çoklu veri saklama altyapısının temelinde "çoklu veri saklama çatısı" bulunmaktadır. Bu çatı içinde, Java Persistence API (JPA) [4] ile uyumlu olarak veri tabanı tabloları

ve uygulama tarafından kullanılan nesneler arasından eşleme yapılabilmekte, açıklama (annotation) veya XML tabanlı yapılandırma ayarları okunup uygulanabilmekte, yazılan sorgular biçimlendirilip işlenebilmektedir.

Çoklu veri saklama çatısı, yapılandırma ayarlarına göre birden fazla veri tabanı bağdaştırıcısı ile etkileşime geçebilmektedir. Veri tabanı bağımlı işlemler bağdaştırıcı bileşenler içine saklanmış, çoklu veri saklama çatısı içindeki işlemler ise tamamen JPA standardına uyumlu olarak geliştirilmiştir.

Çoklu bağdaştırıcı kullanan işlemler, varlıklar içindeki özniteliklere yapılan açıklama (annotation) temelli eşlemelere göre ayrıştırılır ve ilgili veri tabanı bağdaştırıcısına yönlendirilir. Benzer şekilde, yapılan sorgu sonucu birden fazla veri tabanı bağdaştırıcısı üzerinden toplanır ve çoklu veri saklama çatısı altında tek bir nesne içine toplanarak sorgu sonucu olarak sunulur.



Şekil 2. Çoklu Veri Saklama Altyapısı

İstemci uygulama tarafından yapılan bir "persist" isteği, konfigürasyona göre İlişkisel Veri Tabanı veya Cassandra Veri Tabanlarından birisi kullanılarak saklanabildiği gibi (Polystore [7]), "persist" isteği ile gönderilen verinin bir kısmı İlişkisel Veri Tabanı diğer bir kısmı da Cassandra Veri Tabanında saklanabilmektedir (Polyglot [8]).

Benzer şekilde, istemci uygulama tarafından yapılan bir "read" isteği, konfigürasyona göre kullanılan veri tabanlarından birine giderek veriyi çekebildiği gibi (Polystore), parçalı şekilde her iki veri tabanında saklandığı durumlarda her iki veri tabanına birden giderek ilgili kısımları toplamakta ve kullanıcıya farklı veri tabanlarında saklanan kısımları birleştirerek bütün bir nesne yapısı içinde sunmaktadır.(Polyglot).

Şekil 2'de gösterildiği gibi istemci uygulamalar altyapı ile JPA arayüzü üzerinden haberleşmektedirler. Başka bir deyişle istemci uygulama tarafından yapılan tüm veri saklama, okuma ve sorgulama işlemleri JPA kullanılarak standart bir arayüz üzerinden gerçekleştirilmektedir. Çoklu Veri Saklama Altyapısı, JPA üzerinden yapılan bu istekleri önceden yapılmış konfigürasyona göre farklı veri saklama teknikleri kullanarak gerçekleştirmektedir. Bu sayede istemci uygulamalar aynı JPA arayüzlerini kullandıklarından veri saklama yöntemi değişikliklerinden soyutlanmaktadırlar.

4 Örnek Olay Çalışması

KKBS entegrasyonları kapsamında alınan verinin içeriği genellikle iz, izin durumu ve konumu bilgilerini barındırmaktadır. Bu bilgiler incelendiğinde, en sık değişen verinin konum verisi olduğu gözlemlenmiştir. Bu bölümde, duraksız verilerin JC3IEDM

veri modeline kaydedilmesi esnasında darboğaz yaratan iz ve iz konumu ile ilgili veriler temel alınarak bir örnek olay çalışması ortaya konmuştur.

Duraksız iz ve iz konumu verisi sağlayabilmek için TCP-IP protokolü üzerinden veri sağlayacak olan sentetik veri üretici yazılım kullanılmıştır. Duraksız veri, ilk önce ilişkisel veri tabanı, sonrasında ise çoklu veri saklama altyapısı kullanılarak işlenecektir. Çoklu veri saklama altyapısı içinde iz bilgileri ilişkisel veri tabanı, izin konum bilgileri ise sütun temelli veri tabanı üzerinde tutulacak şekilde saklanıp çekilecektir. Her iki durumda da okuma ve yazma süreleri kayıt altına alınacaktır.

4.1 Deney Düzeneği ve Sentetik Veri Üretici

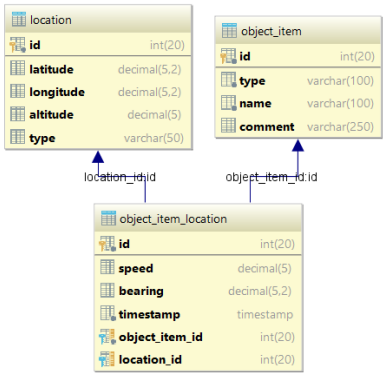
İlgili örnek olay çalışması için, Intel Core i7 işlemci ve 16 GB RAM özellikli donanıma ve Windows 8 işletim sistemine sahip bir çalışma ortamı kullanılmıştır. Sentetik veri üretici uygulaması ve veriyi işleyen uygulamalar aynı ortamda çalıştırılmakta, veri tabanı sunucuları ise aynı özelliklere sahip farklı ortamlarda çalıştırılmaktadırlar.

Simülatör tarafından oluşturulan sentetik veri içeriği yapısı temel iz bilgileri ve iz konumu verilerinden oluşmaktadır.

4.2 RDBMS Örnek Olay Çalışması

RDBMS örnek olay çalışması kapsamında kullanılan veri tabanı modelinin gerçek duruma en yakın sonuçları verebilmesi için JC3IEDM'e uygun bir veri modeli yapısı kullanılmıştır. Buna göre iz bilgilerinin saklanabilmesi için **"object_item"** tablosu, ize ait konum bilgilerinin saklanabilmesi içinse **"location"** ve **"object_item_location"** tabloları kullanılmıştır.

İlişkisel veri tabanı olarak kolay erişilebilir ve açık kaynaklı olmasından dolayı MySQL (5.7.13-log sürümü) [2] kullanılmıştır. Java tabanlı örnek uygulama ve veri tabanı arasındaki bağlantı Spring Data JPA (2.1.3.RELEASE sürümü) ile sağlanmıştır.



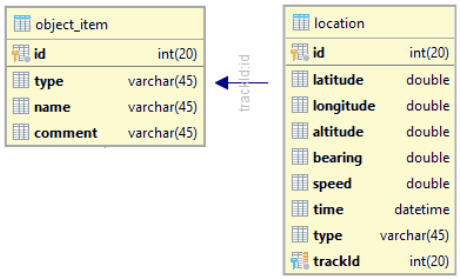
Şekil 3. RDBMS Veri Tabanı Modeli

Sentetik veri üreticiden gelen veriler, ilişkisel veri tabanı için kullanılan veri modeline dönüştürülerek JPA çatısı kullanılarak MySQL veri tabanına saklanmakta ve çekilmektedir. Bu kapsamda alınan sonuçlar bölüm 4.4'de sunulmuştur.

4.3 Çoklu Veri Saklama Altyapısı Örnek Olay Çalışması

Çoklu veri saklama altyapısı örnek olay çalışması kapsamında, hem ilişkisel hem de sütun tabanlı veri tabanları kullanılmıştır. Bu kapsamda ilişkisel veri tabanı olarak bölüm 4.2 ile uyumlu şekilde MySQL (5.7.13-log sürümü), sütun temelli veri tabanı olarak da Apache Cassandra [1] kullanılmıştır. Buna göre temel iz bilgisi ilişkisel veri tabanında saklanırken, duraksız akan konum verisi ise sütun temelli veri tabanı üzerinde saklanacaktır.

Sütun temelli veri tabanı kullanımı için Şekil 3'de belirtilen veri yapısı sadece konum bilgisi için değiştirilerek Şekil 4'teki gibi güncellenmiştir. Buna göre **"object_item"** tablo yapısı yine ilişkisel veri tabanında saklanacağından aynen korunmuş, **"object_item_location"** tablosu ise sütun temelli veri tabanı için kullanımı olmadığından dolayı çıkarılmıştır. Konum ve iz arasındaki ilişki ise **"location"** tablosundaki **"trackId"** alanı üzerinden kurulmuştur.



Şekil 4. Çoklu Model Veri Tabanı Modeli

İki farklı veri tabanı teknolojisi kullanılarak saklanan iki ayrı tablo üzerinde yazma ve okuma işlemlerinin aynı anda gerçekleştirilebilmesi için bir önceki bölümlerde bahsedilen "çoklu veri saklama çatısı" geliştirilmiş, böylece tek bir yazma ve okuma komutu ile ayrı veri tabanları ve ayrı tablolara veri yazılıp okunur hale getirilmiştir. Burada geliştirilen çatı genel olarak farklı veri tabanı yapılarına münferit "veri kaynağı" atanabilmesine ve gerekli işlemlerin bu kaynak üzerinden yapılabilmesine olanak sağlamaktadır. Bu sayede farklı veri tabanı teknolojilerinin güçlü yanlarının aynı anda kullanılabilmesine olanak sağlanmıştır.

4.4 RDBMS ve Çoklu Veri Saklama Altyapısı Örnek Olay Çalışması Karşılaştırması

Veri saklamak için kullanılan çoklu ve ilişkisel veri tabanı çözümünü karşılaştırmak için yazma ve okuma sorguları yürütülmüştür. Kullanılan veri modeline göre bir izin konumu güncellenmek istendiğinde geçmişe yönelik veri saklama zorunluluğundan

dolayı, konum verisi tabloya yeni bir kayıt olarak girilir, var olan değer silinmez ya da güncellenmez. Bu sebeple silme ve güncelleme sorguları kapsam dışı bırakılmıştır.

Karşılaştırma için kullanılan sorgular aşağıda belirtilmiştir:

1000 konum güncelleme: Duraksız veri ile alınan, 1000 konum verisinin aynı anda geldiği durumun yansıtılması amacı ile kullanılmaktadır.

10000 konum güncelleme: Duraksız veri ile alınan, 10000 konum verisinin aynı anda geldiği durumun yansıtılması amacı ile kullanılmaktadır

Tüm izlerin okunması: Duraksız veriden gelen konum verileri göz ardı edilerek sadece ize ait verilerin okunması durumu.

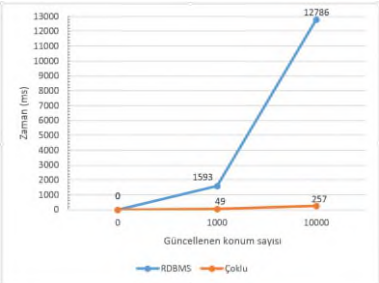
Bir ize ait tüm konumların geçmişe yönelik okunması: Duraksız veri üzerinden gelen konum verileri kaydedildikten sonra, bir ize ait tüm konum verilerinin geçmişe yönelik okunması durumu.

Bir izin en güncel konumunun okunması: Duraksız veri üzerinden gelen konum verileri kaydedildikten sonra, bir ize ait en güncel konum verisinin okunması durumu.

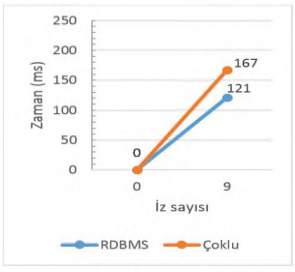
Çoklu veri saklama ve RDBMS çözümlerinin yukarıda bahsedilen sorgular üzerindeki performansları aşağıdaki şekillerde görselleştirilmiştir. Buna göre Şekil 5'te iz konumu güncelleme verileri gösterilmektedir. Çoklu veri saklama altyapısı ile yapılan güncelleme işlemleri ilişkisel veri tabanı üzerinden yapılan güncelleme işlemlerine oranla daha iyi bir performans sergilemektedir. Özellikle anlık güncellenecek konum sayısı arttıkça performans farkının da önemli ölçüde arttığı gözlemlenmiştir.

Şekil 7 kapsamında sadece iz bilgisinin saklandığı durumdaki performans karşılaştırması gösterilmektedir. Bu şekil, çoklu veri saklama yapısının sadece ilişkisel veri tabanı kullandığı durumu göstermektedir. Çoklu saklama altyapısı ile ilişkisel veri tabanı kullanılan durum arasında az bir performans farkı gözlemlenmektedir. Bunun sebebi ise çoklu veri saklama yapısında var olan veri tabanı bağdaştırıcısına yönlendirme işleminden kaynaklanan gecikmedir.

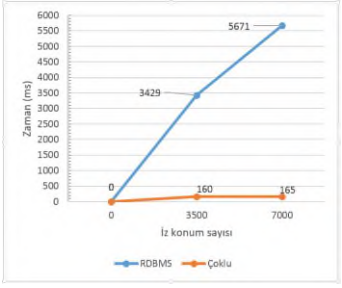
Şekil 6 ve Şekil 8'de ise kaydedilen konum verileri üzerinden yapılan sorgulama işlemlerinin performans karşılaştırılması verilmiştir. Hem güncel konum hem de geçmişe yönelik konum okuma işlemlerinde çoklu veri saklama altyapısı kullanılan durumlarda performans artışı gözlemlenmektedir. Özellikle, var olan iz konum sayısı arttıkça bu performans farkının da arttığı gözlemlenmektedir.



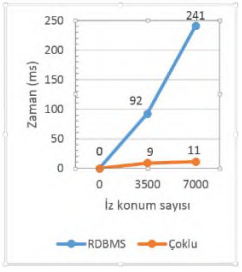
Şekil 5. Konum Yazma



Şekil 6. İze Ait Konumların Okunması (Geçmişe Yönelik)



Şekil 7. İzlerin Okunması



Şekil 8. İze Ait Son Konumun Okunması

5 Sonuç

Yapılan örnek olay çalışması sonucunda, geliştirilen çatı ve oluşturulan çoklu veri saklama yaklaşımının, bileşenlerin güçlü özelliklerini sisteme kazandırma noktasında başarılı olduğu gözlemlenmiştir. Hem ilişkisel veri tabanlarının güçlü özelliklerini muhafaza etmek, hem de ilişkisel olmayan veri tabanlarının güçlü yönlerini sisteme kazandırmak mümkün olmuştur.

Verilen ölçüm süreleri temel alındığında, yazma ve okuma hızlarında yüksek farkların olması beklenen bir sonuçtur. Dolayısıyla aslında hem mevcut sistemlerin bu yapıya evrilmesinin, hem de gelecekte yapılacak çalışmaların bu kapsamda değerlendirilmesinin, bu alanda kronikleşmiş birçok problemin çözülmesine olanak sağlayacağı değerlendirilmektedir.

Kaynakça

1. The Apache Software Foundation: Apache Cassandra, http://cassandra.apache.org/
2. Oracle Corporation: MySQL Enterprise Edition, https://www.mysql.com
3. Polyglot Processing, http://datadventures.ghost.io/2014/07/06/polyglot-processing
4. Oracle Corporation: Java Persistence API, https://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html
5. Wiese, L.: LWA 2015 Workshops: KDML, FGWM, IR, and FGDB. Polyglot Database Architectures. pp. 422-426, [Trier, Germany] (2015).
6. Multilateral Interoperability Programme (MIP) : The Joint C3 Infor-mation Exhange Data Model (JC3IEDM), Greding (2012)
7. Karimov, J., Rabl, T., Markl, V.: Molecular Orientation and Emission Characteristics of Ir Complexes and Exciplex in Organic Thin Films. pp. 24-41, Cham, (2019).
8. Araújo, A., Times, V., Urbano, M.: Int'l Conf. Internet Computing and Internet of Things. PolyEHR: A Fra-mework for Polyglot Persistence of the Electronic Health Record. pp. 71-77, [Las Vegas] (2016).

Yazılım Mühendisliğinde Büyük Veri Analizi: MEASURE – WEKA

Utku Apaydın[1], Selami Bagriyanik[2], Adem Karahoca[3], Mete Aktaş[4], Tolga Ensari[5], Sinan Verdi[6], Ayşe Tosun[7], Hamit Doğan[8]

[1] Bilgi Teknolojileri Bölümü Yüksek Lisans, Bahçeşehir Üniversitesi, İstanbul, Türkiye
utku.apaydin@gmail.com

[2] Dijital Servisler ve Çözümler Teknoloji Bölümü, Turkcell Teknoloji, İstanbul, Türkiye
selami.bagriyanik@turkcell.com.tr

[3] Bilgisayar Mühendisliği, Nişantaşı Üniversitesi, İstanbul, Türkiye
akarahoca@gmail.com

[4]Turkgen Yazılım Sanayi ve Ticaret A.Ş., İstanbul, Türkiye
mete.aktas@turkgen.com.tr

[5]Bilgisayar Mühendisliği, İstanbul Üniversitesi-Cerrahpaşa, İstanbul, Türkiye
ensari@istanbul.edu.tr

[6] Ericsson Araştırma Geliştirme ve Bilişim Hizmetleri A.Ş., İstanbul, Türkiye
sinan.verdi@ericsson.com

[7]Bilgisayar ve Bilişim Fakültesi, İstanbul Teknik Üniversitesi, İstanbul, Türkiye
tosunay@itu.edu.tr

[8]Turkgen Yazılım Sanayi ve Ticaret A.Ş., İstanbul, Türkiye
hamitdogan17@gmail.com

Özet. Günümüzde yazılım mühendisliğinin çözmesi beklenen en önemli sorunlardan birisi, zaman baskısı altında kaliteli yazılım ürün ve çözümlerinin hayata geçirilmesi olmaya devam etmektedir. Bu ihtiyaçları karşılamak üzere, yazılım geliştirme yaşam döngüsü içinde üretilen büyük verinin metriklere dönüştürülmesi, ölçülmesi; veri madenciliği ve makine öğrenmesi yöntemleri ile bütünsel olarak gerçek zamanlı veya toplu olarak analiz edilmesine yönelik çalışmalar da hem endüstri hem de akademik dünyada yer bulmaya başlamıştır. Bu çalışma kapsamında, yukarıda sözü edilen yazılım mühendisliği büyük verisinin tek platformda toplanması ve ölçüm metriklerinin oluşturulmasını hedefleyen ITEA girişimlerinden MEASURE platformuna makine öğrenmesi yeteneklerinin kazandırılmasının ilk aşaması gerçekleştirilmiştir. Bu kapsamda, MEASURE platformu ve WEKA makine öğrenmesi yazılımı arasındaki entegrasyon işleminin nasıl gerçekleştirildiği ve WEKA yazılımı ile oluşturulan veri analizi sonuçlarının MEASURE platformuna nasıl aktarıldığı açıklanmıştır. Oluşturulan bu yapı sayesinde gelecek araştırmamız kapsamında olan

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

MEASURE platformu üzerindeki veriler kullanılarak oluşturulacak öngörü modellerini destekleyen bir alt yapı elde edilmiştir. Geliştirilen entegrasyonu doğrulamak amacıyla daha önce yapılan bir gereksinim mühendisliği araştırmasında kullanılan eğitsel veriye çeşitli makine öğrenmesi yöntemleri uygulanmış, öngörü performansları karşılaştırılmış ve bir önceki çalışma ile uyumlu olduğu gözlenmiştir.

Anahtar Kelimeler: Yazılım Mühendisliği, Makine Öğrenmesi, Büyük Veri, Yazılım Mimarisi ve Analizleri, Yazılım Hayat Döngüsü, MEASURE platformu, WEKA

Big Data Analysis in Software Engineering: MEASURE – WEKA Integration

Utku Apaydin[1], Selami Bagriyanik[2], Adem Karahoca[3], Mete Aktas[4], Tolga Ensari[5], Sinan Verdi[6], Ayse Tosun[7], Hamit Dogan[8]

[1] Information Technologies Department, Bahcesehir University, Istanbul, Turkey
utku.apaydin@gmail.com

[2] Digital Services and Solutions Technology Department, Turkcell Technology, Istanbul, Turkey
selami.bagriyanik@turkcell.com.tr

[3] Computer Engineering Department, Nisantasi University, İstanbul, Türkiye
akarahoca@gmail.com

[4]Turkgen Yazılım Sanayi ve Ticaret A.Ş., Istanbul, Turkey
mete.aktas@turkgen.com.tr

[5]Computer Engineering Department, Istanbul University-Cerrahpasa, Istanbul, Turkey
ensari@istanbul.edu.tr

[6] Ericsson Araştırma Geliştirme ve Bilişim Hizmetleri A.Ş., Istanbul, Turkey
sinan.verdi@ericsson.com

[7]Faculty of Computer and Informatics Engineering, Istanbul Technical University, Istanbul, Turkey
tosunay@itu.edu.tr

[8]Turkgen Yazılım Sanayi ve Ticaret A.Ş., Istanbul, Turkey
hamitdogan17@gmail.com

Abstract. Nowadays, one of the most challenging problems in software development has long been the realization of high-quality software products and solutions under time pressure. In order to meet these needs, capturing metrics from within big data which is generated through software development life cycle and analyzing the data using machine learning / data mining methods started to become popular among companies and research institutions. Within the scope of this study, the first stage of bringing in machine learning capabilities to the MEASURE platform, one of the ITEA initiatives aiming at gathering the above-mentioned software engineering big data in a single platform and creating measurement metrics, has been realized. In this context, the integration process between MEASURE platform and WEKA machine learning software and how data analysis results are transferred to MEASURE platform have been explained. Thanks to this structure, a mechanism that enables the prediction models to be created by using data on the MEASURE platform which is within the scope of our future research, has been obtained. In order to validate the integration, we applied several machine learning algorithms using an educational data obtained from a previous requirement engineering research, compared their accuracy results and observed that the results are in agreement.

Keywords: Software Engineering, Machine Learning, Big Data, Software Architecture and Analytics, Software Development Life Cycle, MEASURE platform, WEKA

1 Giriş

Yazılım mühendisliği, günümüz teknolojisinde üzerinde pek çok çalışma yapılan en geniş çalışma ve araştırma alanlarından biri olarak karşımıza çıkmaktadır. Yakın geçmişteki ilgili literatür incelendiğinde, özellikle makine öğrenmesi ve büyük veri kapsamındaki katkıların artmaya başladığı görülmektedir. Bu bilimsel çalışmaların gerçekleştirilmesi için toplanan bilgilerin temiz ve düzenli bir yapıya sahip olması yapılacak olan çalışmanın başarısını büyük oranda etkilemektedir [2, 14]. Yazılım endüstrisi içinde, Yazılım Geliştirme Hayat Döngüsü (SDLC: Software Development Life Cycle) yönetimi için kullanılan araçlar (Jira, ALM, VersinOne, Polarion vb.) verimlilik artışlarına katkı sağlamıştır [17, 20]. Bu araçlar analiz, test ve geliştirme sırasında yazılım endüstrisi çalışanlarının aktif olarak kullanması sonucu düzenli ve ölçülebilir veriler oluşturmaktadır. Bununla birlikte bu veriler, toplandıkları araçların ihtiyaçları doğrultusunda modellenmiş olmaları nedeniyle üzerlerinde öngörü yapılabilmesi için ihtiyaç duyulan ölçümleme model yapısına dönüştürülmesi gerekmektedir (II. bölümde bu model yapısı açıklanmıştır). İhtiyaç duyulan bu tür dönüşümler sırasında verilerin alanında uzman olan çalışanlar tarafından tutarlı ve tutarsız olarak ayrıştırılması gerekmektedir [3]. Bu çalışma kapsamında, WEKA makine öğrenmesi yazılımının MEASURE platformu ile nasıl entegre edildiği açıklanmaktadır. Gerçekleştirilen entegrasyonun doğrulanması (validasyonu) için, alanında lider büyük ölçekli bir teknoloji ve hizmet sağlayıcı şirketin bünyesinde toplanan veriler ile analist pozisyonundaki çalışanların eğitim ihtiyaçlarını öngören daha önce aynı firmada yapılmış bir çalışmadaki [1] model kullanılmıştır. Oluşturulan

bu altyapı sayesinde MEASURE [10] projesi kapsamında gerçekleştirilen test otomasyon senaryolarının adımlarının koşturulması sırasında üretilen verileri toplayıp veri analizi yapacak altyapı bu çalışma ile sağlanmıştır. Bu entegrasyon sayesinde daha önce bahsedilen öngörü verilerinin toplanması, uygun model yapısına çevirilerinin gerçekleştirilmesi ve kalitesiz veri gruplarının elenmesini sağlayan bir yapı elde edilmiştir [2]. Literatürdeki konuyla ilgili çalışmalar incelendiğinde, büyük veri işleme aracı olarak kullanılan makine öğrenmesi yöntemleri üzerinde çalışıldığı görülmektedir [4, 5, 6]. Öğrenme tabanlı bu sistemlerle, gelişmeler kaydedilmiştir [7,12,13,16]. Bu sistemlerin özellikle akademik araştırmalarda en yaygın kullanılanlarından biri olan WEKA yazılımı [19] bu çalışmada ana kütüphane olarak kullanılmıştır. Yukarıda bahsedilen temiz ve güvenilir verilerin bir noktada toplama ihtiyacı MEASURE platformunun sağladığı altyapı kullanılarak giderilmiştir. Sonraki bölümlerde, sırası ile WEKA yazılımının tanıtılması, MEASURE platformunun tanıtılması, iki bileşen arasındaki entegrasyon, bu entegrasyon aracılığıyla yapılan öngörülerin MEASURE platformuna kaydedilmesi ve ilerideki araştırma konumuz olan MEASURE platformuna kaydedilen veriler ile öngörülerde bulunmamızı sağlayacak yapının nasıl bir tasarıma sahip olacağını açıklamaktadır.

2 Makine öğrenmesi ve weka

WEKA (Waikato Environment for Knowledge Analysis), gerçek dünya verileriyle makine öğrenmesi tekniklerinin kullanımına olanak sağlayan bir yazılımdır [19]. Bu yazılım makine öğrenmesi, nitelik seçimi, sınıflandırma, kümeleme gibi birçok özelliğe sahiptir [18]. Kütüphane bünyesinde bulunan onlarca makine öğrenmesi tekniğini, öngörü için kullanılacak verileri ARFF (Attribute-Relation File Format) formatındaki dosyalardan okuyarak sahip olduğu Java tabanlı ara yüz aracılığı ile gerçekleştirebilmektedir. ARFF veri formatında (Şekil 1) veri noktaları (data point) kendi arasında sayısal (numeric) ve sembolik (nominal) olmak üzere temel iki sınıfa ayrılmıştır. Bu formatta en sonda yer alan veri özelliği, sınıf (class) olarak isimlendirilir. Kütüphane geçmiş veriler üzerinden eğitilerek, bu eğitim yüzdesel bölme (percentage split) veya çapraz validasyon (crossvalidation) yöntemlerinden biri kullanılarak yapılmaktadır. Böylece daha sonra, sınıf bilgisi verilmemiş veri kümeleri için öngörülerde bulunabilmektedir.

```
@relation cfp_edu

@attribute usecse {YES,NO,PARTIAL}
@attribute interaction {YES,NO,PARTIAL}
@attribute information {YES,NO,PARTIAL}
@attribute business_logic {YES,NO,PARTIAL}
@attribute applicaiton {YES,NO,PARTIAL}
@attribute scope {YES,NO,PARTIAL}
@attribute cfp_training_need {YES,NO}

@data
YES,YES,YES,YES,YES,YES,NO
YES,YES,PARTIAL,YES,YES,YES,YES
YES,YES,PARTIAL,PARTIAL,YES,YES,YES
YES,YES,YES,YES,YES,YES,NO
YES,YES,YES,YES,YES,YES,NO
YES,YES,PARTIAL,YES,YES,YES,NO
```

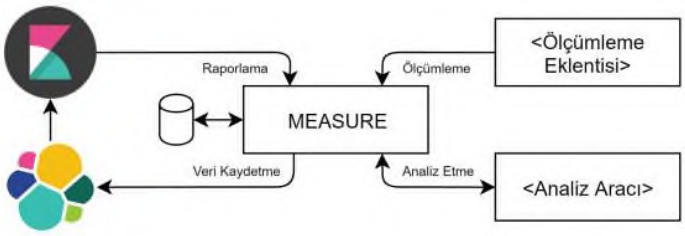
Şekil 1. ARFF Veri Formatı

İşaret işleme tabanlı, istatistik tabanlı veya makine öğrenmesi tabanlı sistemler sayesinde, akıllı sistemlerin gerçeklenmesi yönünde önemli gelişmeler kaydedilmiştir [1-4]. Bu bağlamda, makina ögrenmesi çalışması için verilerini dijital ortamda taşıyan tüm sistemler için benzer örüntü tanıma işlemleri gerçeklenerek akıllı sistem oluşturularak, gelişmeler sağlanabilir. Literatürde yapılan çalışmalar incelendiğinde, boyut azaltmada kullanılan temel bileşen analizi (principal component analysis), merkez/uzaklık metrik tabanlı k-ortalamalar (k-means), J48(C4.5 tekniğinin Java ile gerçeklenmiş halidir.) versiyonlarına sahip karar ağaçları (decision trees), Bayes olasılık teorisini kullanan Naive Bayes, insan sinir ağını örnek alarak oluşturulan yapay sinir ağları (aritificial neural networks), karar sınırları/hiper düzlemleri arasında sınıflandırma yapan karar destek vektörleri (support vector machines) ve en yakın k adet komşuluk değerine göre sınıflandırma yapan k-en yakın komşuluk (k-nearest neighbour) algoritması, kullanılabilecek makine öğrenmesi yöntemleri olarak karşımıza çıkmaktadır [8, 9, 11, 15]. Bu yöntemlerin hepsi WEKA yazılımı tarafından desteklenmekte ve MEASURE'da kurulan altyapı temiz ve merkezi verilerin bu çalışma kapsamı dışındaki modellerle de öngörü modellerinin kurulmasına olanak sağlamaktadır.

3 Meaure platformu

Bu çalışma kapsamında entegre olunan, öngörüleri ve/veya bu öngörüler için gerekli ölçümleme verilerini saklayan MEASURE platformu, içinde Türkiye konsorsiyumunun da yer aldığı uluslararası bir yazılım Ar-Ge projesidir [10]. MEASURE platformu; verilerin toplanması, bununla ilgili veri toplama zincirlerinin (akışlarının) oluşturulması, gelişim süreçlerinin tanımlanması, kalite araç kutusunun

oluşturulması ile ön tahmin baraj değerlerinin bilinmesi/tutturulması sayesinde yazılım ve test mühendisliği gelişimi yönüyle bütünsel kalitenin arttırılması için çeşitli yaklaşımların sunulduğu çalışmaları içermektedir. Bu platformun amacı, Avrupa'da yazılım mühendisliği maliyetlerini ve piyasaya sürülme zamanını düşürmenin yanı sıra kaliteyi ve verimliliği arttırmaktır. Platform bu amacını kendisine bağlanan servisleri, dışarı açtığı uygulama programlama arayüzü (API: Application Prgramming Interface) ve jenerik hale getirilmiş veri modellerini kullanarak, Elastic Search ve Kibana [21] üzerinden verileri görselleştirerek sağlayabilmektedir. Dinamik olarak düzenlenebilen rapor paneli sayesinde kullanıcılar platform üzerinde oluşturdukları projeler ile istedikleri veri kümesi üzerinden planlı ya da isteğe bağlı olarak elle raporlar oluşturabilmektedir. MEASURE platformu açıkladığımız özellikleri var olan dokümanlardan [10] görüldüğü kadarıyla, aşağıda gösterilen (Şekil 2) yapı sayesinde gerçekleştirebilmektedir.



Şekil 2. MEASURE platformu

Şekil 2 de belirtildiği gibi MEASURE platformu kendisine sunulan verileri Elastic Search altyapısının sağladığı indexleme yöntemleri ile saklamaktadır. Bu verilerin içeri alınması, MESURE arayüzleri üzerinden ya da platformun açtığı API'lar üzerinden tetiklenebilmektedir. Tetikleme sonrasında verileri MEASURE platformu tanımlanmış olan entegrasyonlar üzerinden almaktadır, dış bir systemin platforma veri göndermesi engellenmiştir. Bu kısıtlar sayesinde aşağıda detaylandırılan adımlardan Analiz etme ve Ölçümleme işlemleri ayrı sistemler üzerinden gerçekleştirilebilmektedir. Jenerik bir yapı kurulması durumunda MEASURE üzerindeki tüm veriler ile tahminleme çalışmaları yapılması mümkündür. Bu çalışma ise bu ihtiyacı gidermek için gerçekleştirilmiştir.

Raporlama; MEASURE rapor paneli üzerindeki çıktılarını Kibana aracılığı ile oluşturmaktadır.

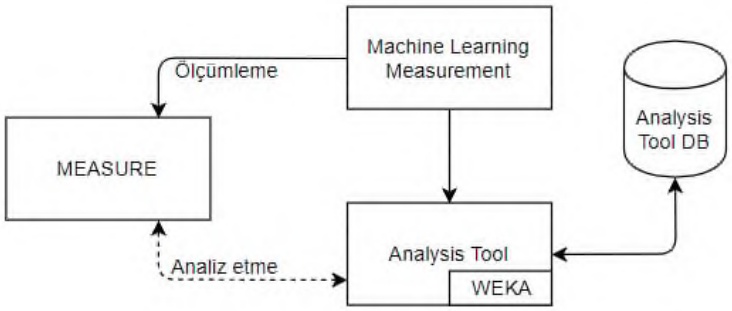
Veri kaydetme; Platform üzerine kaydedilen tüm ölçümlemeler ElasticSearch üzerinde tutulmaktadır.

Ölçümleme; Platformun kendi bünyesine jenerik bilgi kaydetmesini sağlayan yapıdır. Dış dünyadan konu ve içerik bağımsız gerçekleştirilen ölçümleme sonuçların bu yapı ile ElasticSearch üzerine aktarılmaktadır.

Analiz Etme; Platformun üzerindeki verilerin dış dünyadaki herhangi bir sistem tarafından okunmasını bu veriler üzerinde işlem yapılmasını ve sonuçlarının tekrar MEASURE platformuna aktarılmasını sağlayan yapıdır.

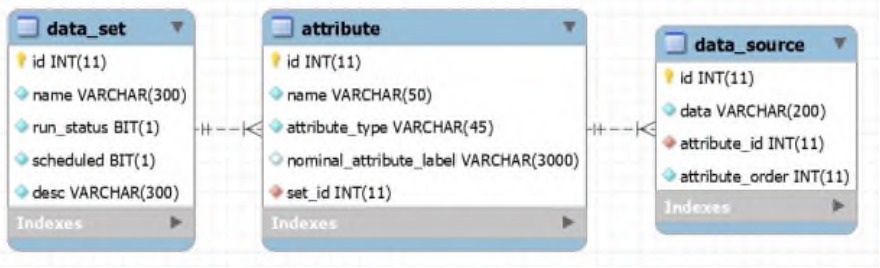
4 Measure – weka Entegrasyonu

Bu çalışma kapsamında bir önceki bölümde anlatılan Ölçümleme akışına dahil olacak bir entegrasyon gerçekleştirilmiştir. Ölçümleme işlemini gerçekleştirirken WEKA yazılımının üzerinde bulunan makine öğrenmesi algoritmaları kullanılmaktadır. Ölçümleme işlemi sonundaki çıktılar ML Measurement (Şekil 3 de gösterilen Machine Learning Measurement) üzerinden MEASURE platformunda bulunan Elastic Search yapısına kaydedilmektedir. Geliştirilen Analiz aracı WEKA mimarisinde ihtiyaç duyulan ARFF model yapısına dönüştürme işlemini otomatik gerçekleştirmektedir. Ölçümleme yapılacak veri kümeleri ise uygulamanın kendi veri tabanında (Şekil 3 de gösterilen Analysis Tool DB) Şekil 4'te gösterilen varlık-ilişki veri modeline uygun şekilde MYSQL [22] veri tabanında tutulmaktadır.



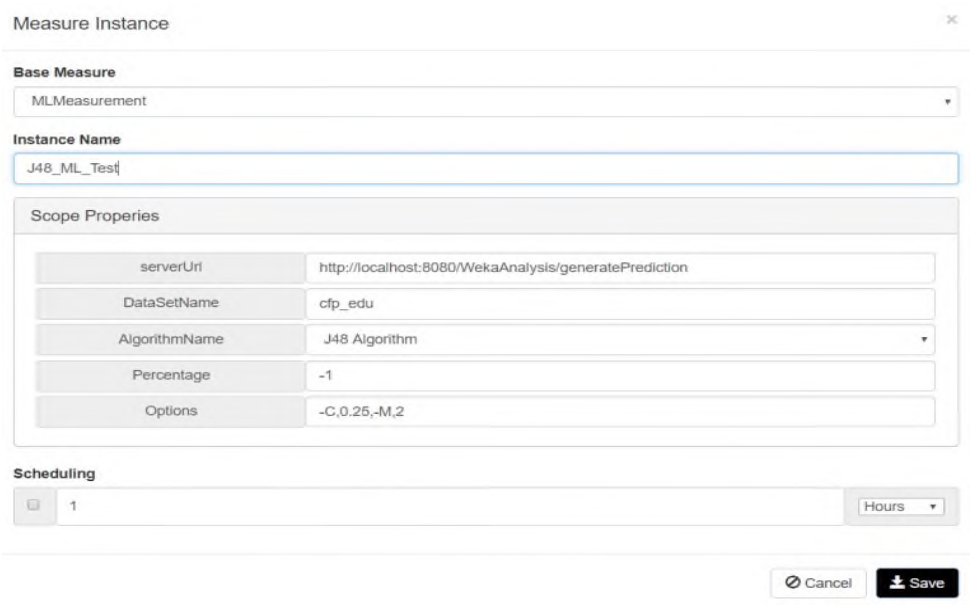
Şekil 3. MEASURE ara yüzü için yazılım test ve makine öğrenmesi analiz aracı

Analiz aracı arka planda bir Apache server üzerinde çalışmakta ve verilere Hibernate kütüphanesini kullanarak erişmektedir. Analiz aracının bir arayüz modülü bulunmamaktadır. Uygulama sadece bir servis sağlayıcı olarak görev yapmaktadır. ARFF dosyalarındaki bilgiler manuel olarak veri ekleme sorgularına çevrilerek veri tabanına kaydedilmektedir. Mevcut durumda analiz aracı 0.1.6 versiyonu ile 3 ayrı algoritmayı desteklemektedir. Bunlar sırası ile J48, Naive Bayes ve Çok katmanlı Algılayıcı (MLP) teknikleridir. Analiz aracı öngörülerini yapabilmek için WEKA yazılımına ihtiyaç duyduğu parametreleri MEASURE ara yüzü aracılığı ile kullanıcılardan alarak iletmektedir.



Şekil 4. Veri kümesi, öznitelik ve veri kaynağı yapıları

MEASURE platformunun sunduğu IDirect Measure interface tanımı sayesinde dışarıda bulunan servisleri çağırabilmektedir. Platformun mimarisinde bulunan kurallar çerçevesinde bu servislere gönderilebilecek olan parametreler ML Measurement paketi içinde bulunan bir XML tanımı ile tanımlanmaktadır. Bu XML tanımı her bir ölçümleme aracı için özel olarak hazırlanmak zorundadır. Platform hazırlanan XML sayesinde ölçümleme aracı üzerinden gönderilebilecek parametreleri ve geriye dönecek sonucun modelini tespit edebilmektedir. Ölçümleme aracı mimari kısıtlardan dolayı veri tabanı bağlantısı gerçekleştirememektedir. Bu ihtiyacı gidermek için Şekil 3'de gösterilen ve daha önceden bahsettiğimiz analiz aracı geliştirilmiştir. Ölçümleme aracı Analiz aracına MEASURE platformu üzerinden istek gönderilmesini ve sonucun alınabilmesini sağlayan bir eklenti görevi görmektedir. MEASURE üzerinden bir ölçümleme aracı tanımlamak için Şekil 5'te gösterilen ara yüz kullanılmaktadır. Şekilde görünen "Scope Properties" bölümü XML üzerinden tanımlanan MEASURE ara yüzünden yollanabilecek parametrelerin dinamik olarak listelendiği bölümdür. Burada parametre değerleri, XML üzerinden mevcut değerleri tanımlanmış olarak veya belirli tanımlanmış değerler üzerinden seçim yapılabilir açılır liste olarak gösterilmektedir. "Base Measure" olarak isimlendirilmiş alan Ölçümleme aracımızın XML üzerinde gösterilen isminin göründüğü alandır. MEASURE platformunda oluşturulan ölçümleme projesine eklenen bu ölçümleme araçları proje içindeki bilgi kaynakları bölümünde listelenmektedir. İstek dahilinde bu ölçümleme araçları planlı olarak çalıştırılabilir. Ölçümleme sonuçları Elastik Search modülüne kaydedilmektedir.



Şekil 5. MEASURE Ölçümleme Aracı

Ölçümleme araçları üzerinden gerçekleştirilen başarılı çağrıların sonuçları Ölçümleme projesi üzerinde gösterge tablosu üzerinden incelenebilmektedir. Şekil 6'te gösterilen değerler WEKA yazılımı üzerinden örnek diyabet verileri ile çağırılan sırası ile MLP, Naive Bayes ve J48 ölçümlerinin sonuçlarıdır.



Şekil 6. MLP, Naive Bayes ve J48 sınıflandırma sonuçları

Bu çalışma kapsamında diyabet örneği dışında gerçek dünyadan beslenerek yapılan bir öngörü denemesinin sonuçlarını geliştirilen ML Measurement üzerinden tekrar elde edilmiştir. Bu öngörü, denemesi analiz dokümanlarından elde edilen veriler ile çalışanlar üzerinde CFP eğitimi ihtiyacını öngören model esas alınarak gerçekleştirilmiştir [1].

5 Sonuçlar

Bu çalışma kapsamında, tahminleme çalışması yapılmak istenen ilgili yazılım ortamlarından toplanan veriler ile gerçekleştirilen öngörü işlemleri geliştirilen analiz aracı ile yapılabilir duruma getirilmiştir. Elde edilen sonuçlar, MEASURE platformu sayesinde merkezi bir sistem üzerinde saklanıp raporlanabilir durumdadır.

Kurulan yapının beklentiyi karşıladığını teyit etmek için ilgili ortamda gerçekleştirilmiş olan eğitim ihtiyaçlarını öngören çalışma [1] verisi ve bu sonuçları kullanarak aşağıdaki sonuçlar elde edilmiştir. İlgili çalışmadaki sonuçlar aşağıdaki gibidir. Referans aldığımız çalışmanın [1] çıktıları ile bu makalede anlatılan entegrasyon çalışması sonucunda elde edilen çıktılar (Tablo 1) arasında tespit edilen sapmalar WEKA kütüphanesinden [23] kaynaklı sapmalardır, bu sapmaların nedeni her bir weka kurulumunun kendisine ait bir tohum (seed) tanımı bulunması ve bu tanımın model eğitimi sırasında kullanılmasıdır.

Tablo 1. CFP Ontoloji Gösterge seti ve Olası Değerleri ve Entegrasyon Çıktıları

	Orijinal Makale Çıktıları [1]						Entegrasyon Sonuçları			
	Deney 1: Çapraz - validation			Deney 2: 66% Split			Çapraz - Validation		66% Split	
Algoritma	**Doğruluk (%)**	**FM**	**Kappa**	**Doğruluk (%)**	**FM**	**Kappa**	**Doğruluk(%)**	**Kappa**	**Doğruluk(%)**	**Kappa**
Random Forest (RF)	78.79 (11.72)	0.82 (0.11)	0.56 (0.25)	79.20 (5.54)	0.83 (0.04)	0.56 (0.12)				
Naive Bayes (NB)	82.97 (11.13)	0.86 (0.09)	0.63 (0.24)	81.29 (5.46)	0.85 (0.04)	0.60 (0.12)	83.16	0.64	84.84	0.68
REPTree	84.02 (11.25)	0.86 (0.11)	0.67 (0.23)	84.27 (4.81)	0.86 (0.04)	0.68 (0.10)				
J48 (WEKA C 4.5 uyarlaması)	82.59 (11.39)	0.84 (0.11)	0.65 (0.23)	83.38 (4.58)	0.85 (0.04)	0.66 (0.09)	86.13	0.72	89.39	0.78
Logistic Model Trees (LMT)	83.84 (11.34)	0.86 (0.11)	0.67 (0.23)	83.52 (5.21)	0.86 (0.05)	0.66 (0.11)				
Multilayer Perceptron (MLP)	76.74 (12.50)	0.80 (0.12)	0.52 (0.26)	75.94 (6.55)	0.80 (0.05)	0.49 (0.15)	80.59	0.59	83.39	0.62
Support Vector Machines with SMO	84.16 (11.25)	0.86 (0.11)	0.68 (0.23)	83.70 (4.91)	0.86 (0.04)	0.67 (0.10)				
K-Nearest Neighbour Classifier (IBK)	75.61 (11.56)	0.80 (0.10)	0.48 (0.25)	75.38 (5.16)	0.81 (0.04)	0.47 (0.12)				
OneR	84.16 (11.25)	0.86 (0.11)	0.68 (0.23)	84.27 (4.81)	0.86 (0.04)	0.68 (0.10)				
ZeroR	59.45 (1.64)	0.75 (0.01)	0.00 (0.00)	59.37 (0.63)	0.75 (0.04)	0.00 (0.00)				

Çalışmamız kapsamında, makine öğrenmesi metotlarından J48, MLP ve Naive Bayes yöntemleriyle Tablo 1'in sağ tarafında gösterilen sonuçlar elde edilmiştir.

Analiz aracı 4. bölümde bahsedilen özellikleri sayesinde WEKA kütüphanesindeki ML algoritmalarının çalıştırılabilmesi için gerekli ARFF format dönüşümünü Şekil 4'te belirtilen ER yapısı üzerinden gerçekleştirebilmektedir. Ayrıca bu çalışma ile ARFF formatında olmayan verilerin de MEASURE platformundan çekilerek analiz edilmesini sağlayacak alt yapı kurulmuştur.

Gelecek araştırma konusu olarak, çalışmamızın 3. bölümünde aktarılan, MEASURE platformu mimarisi tarafından desteklenen analiz aracının geliştirilmesi planlanmaktadır. Bu aracın amacı, MEASURE platformu üzerinde bulunan SDLC süreçlerine ait daha fazla veri alarak öngörüler elde etmek ve platforma geri beslemek olacaktır. Bu çalışma kapsamında geliştirdiğimiz MEASURE-WEKA entegrasyonu, MEASURE projesinin Türkiye konsorsiyumunun hedefi olan test otomasyon çalışmasının gerçekleştirilmesi için baz teşkil edecektir.

6 Teşekkür

Bu çalışma, TÜBİTAK 1509 programı tarafından (Proje No: 9170039) desteklenmiştir. TÜBİTAK'a, vermiş olduğu desteklerden dolayı teşekkürlerimiz sunarız.

Kaynakça

1. Bagriyanik, S. & Karahoca, A.: Using Data Mining to Identify COSMIC Function Point Measurement Competence. International Journal of Electrical and Computer Engineering. 8, pp. 5253-5259 (2018).
2. Merino J., Caballero I., Rivas B., Serrano M., Piattini M.: A Data Quality in Use model for Big Data. Future Generation Computer Systems, vol. 63, 2016, pp. 123-130 (2015).
3. Bagriyanik S., Karahoca A.: Big Data in Software Engineering: A Systematic Literature Review, Global Journal of Information Technology, 6 (1), 107-116, (2016).
4. Emilia G. D., Gasbarro D. D., Gaspari A., Emanuela N.: Managing the Uncertainty of Conformity Assessment in Environmental Testing by Machine Learning, 124, 560-567, (2018).
5. Lenz A. R., Pozo A., Vergilio S. R.: Linking Software Testing Results with a Machine Learning Approach, Engineering Applications of Artificial Intelligence, 26, 1631-1640, (2013).
6. Paduraru C., Melemciuc M., Stefanescu A.: A Distributed Implementation Using Apache Spark of a Genetic Algorithm Applied to Test Data Generation, Proc. of the ACM, (2017).
7. Dahab S., Maag S., Bagnato A., Da Silva M. A. A.: A Learning Based Approach for Green Software Measurements", Proc. of the Int. Workshop on Measurement and Metrics for Green and Sustainable Software Systems, 13-22, (2016).
8. Zhang D., Tsaij J. P.: Advances in Machine Learning Applications in Software Engineering, IGI Global Pub., USA, (2007).
9. Challagula V. B., Bastani F. B., Yen I., Raymond A. P.: Empirical Assessment of Machine Learning Based Software Defect Prediction Techniques, Int. Journal of Artificial Intelligence, 17 (2), 389-400, (2008).
10. ITEA 3 · Project · 14009 MEASURE, https://itea3.org/project/measure.html

11. Srinivasan K., Fisher D.: Machine Learning Approaches to Estimating Software Development Effort, IEEE Transactions on Software Engineering, vol. 21, no. 2, (1995).
12. Zhang D., Tsai J.J.P.: Machine Learning and Software Engineering. Software Quality Journal 11, 87–119, (2003).
13. Cui L., Yu F. R., Yan Q.: When Big Data Meets Software-Defined Networking: SDN for Big Data and Big Data for SDN. IEEE Network, 30 (1), (2016).
14. Mockus A.: Engineering Big Data Solutions. FOSE 2014 Proceedings of the on Future of Software Engineering, 85-99, (2014).
15. Madhavji N. H., Miranskyy A., Kontogiannis K.: Big Picture of Big Data Software Engineering: With Example Research Challenges. Proc. of the Int. Workshop on Big Data Software Engineering, (2015).
16. Gondra I., Applying machine learning to software fault-proneness prediction, The Journal of Systems and Software 81, 186–195, (2008).
17. Anderson K. M., Embrace the challenges: Software Engineering in a Big Data World, Proc. of the Int. Workshop on Big Data Software Engineering, pp. 19-25, (2015)
18. Frank E., Hall M. A., Witten I. H.: The WEKA Workbench. Online Appendix for Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, (2016).
19. Hall M., Frank E., Holmes G., Pfahringer B., Reutemann P., Witten I. H. The WEKA Data Mining Software: An Update. SIGKDD Explorations, Vol. 11, no. 1 (2009).
20. Subramanian G. H., Zarnich G. E.: An Examination of Some Software Development Effort and Productivity Determinants in ICASE Tool Projects, Journal of Management Information Systems, 12:4, 143-160 (1996).
21. Open Source Search & Analytics · Elasticsearch | Elastic, https://www.elastic.co/
22. MySQL, https://www.mysql.com/
23. Weka 3 - Data Mining with Open Source Machine Learning Software in Java, https://www.cs.waikato.ac.nz/ml/weka/

Çeviklik Değerlendirme (AgilityMod) ve Süreç Yetenek Değerlendirme (SPICE) Modelleri Saha Uygulamaları

Damla Yaşar[1][0000−0002−9430−6849], Serhat Caner[1], Hüseyin Ünlü[1], Ali Yıldız[2], Arif Kürşat Karabayır[1], Barış Sokat[1], Burak Bilgin[1], Gizem Arık[1], Levent Anıl Özen[1], Osman Sönmeztürk[1], Seren Seyvan[1], and Onur Demirörs[1][0000−0001−6601−3937]

Bilgisayar Mühendisliği Bölümü, İzmir Yüksek Teknoloji Enstitüsü, İzmir, Türkiye
{damlayasar, serhatcaner, huseyinunlu, arifkarabayir, barissokat, burakbilgin, zekiyearik, osmansonmezturk, serenseyvan, onurdemirors}@iyte.edu.tr
Bilgi Grubu, İzmir, Turkey
{ali.yildiz}@bg.com.tr

Özet. Yazılım süreci olgunluğu değerlendirme ve iyileştirme modelleri 1990'lardan bu yana birçok yazılım organizasyonu tarafından kullanılmaktadır. Bu modeller içerisinde en sık kullanılanlar Yetenek Olgunluk Modeli (CMM) ve ISO 15504 / ISO 3300X Yazılım Süreç İyileştirme ve Yetenek Değerlendirme (SPICE) Modelidir. Çevik (Agile) yazılım geliştirme pratiklerinin bu yaklaşımlarla, uygun bir şekilde iyileştirilip iyileştirilemeyeceği ve var olan yaklaşımlardan hangilerinin günümüz organizasyonlarına daha uygun olduğu hakkında belirgin fikir ayrılıkları ve tartışmalar mevcuttur. Bu problemi ele almak için İzmir Yüksek Teknoloji Enstitüsü Bilgisayar Mühendisliği CENG515 dersi kapsamında yüksek lisans ve doktora öğrencileri tarafından çalıştıkları kurumlarda bir proje ile ilgili hem ISO 15504 hem de AgilityMod yaklaşımları kullanılarak ayrı ayrı değerlendirmeler yapılmıştır. Toplamda beş takım değerlendirmelerini tamamladıktan sonra, sonuçlar analiz edilmiş ve hangi organizasyonlar için hangi yaklaşımın daha uygun sonuçlar verdiğine dair incelemeler yapılmıştır. Elde edilen sonuçlara göre, dört durum için AgilityMod; süreçlerin uygunluğu ve elde edilen sonuçların şirkete olan yararı yönünden daha uygun bulunmuştur. AgilityMod'u daha uygun bulan dört durumda incelenen şirketler çevik (Agile) yazılım geliştirme süreci, uygun bulunmayan şirkette ise şelale (Waterfall) yazılım geliştirme süreci kullanıldığı görülmüştür.

Anahtar Kelimeler: Yazılım Süreç Olgunluğu, Yazılım Kalitesi, Süreç İyileştirme.

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Agility Assessment (AgilityMod) and Process Capability Assessment (SPICE) Models Field Applications

Damla Yaşar[1][0000−0002−9430−6849], Serhat Caner[1], Hüseyin Ünlü[1], Ali Yıldız[2], Arif Kürşat Karabayır[1], Barış Sokat[1], Burak Bilgin[1], Gizem Arık[1], Levent Anıl Özen[1], Osman Sönmeztürk[1], Seren Seyvan[1], and Onur Demirörs[1][0000−0001−6601−3937]

Department of Computer Engineering, Izmir Institute of Technology, İzmir, Turkey
{damlayasar, serhatcaner, huseyinunlu, arifkarabayir, barissokat, burakbilgin, zekiyearik, osmansonmezturk, serenseyvan, onurdemirors}@iyte.edu.tr
Bilgi Grubu, İzmir, Turkey
{ali.yildiz}@bg.com.tr

Abstract. Software process maturity assessment and improvement models have been used by many software organizations since the 1990s. The most commonly used models are Capability Maturity Model (CMM) and ISO 15504 / ISO 3300X Software Process Improvement and Ability Assessment (SPICE) Model. There are clear disagreements and discussions about whether Agile software development processes can be improved with these approaches and which of the existing approaches are more suitable for today's organizations. In order to tackle this problem, software process assessments were carried out separately using both ISO 15504 and AgilityMod approaches by the graduate and doctoral students within the scope of CENG515 course of Computer Engineering of İzmir Institute of Technology. After completing a total of five team assessments, the results were analyzed and examinations were conducted on which approaches gave more appropriate results for which organizations. According to the results obtained, for four cases, AgilityMod was found more appropriate for the suitability of the processes and the benefit of the results to the company. It was observed that the companies which found AgilityMod more suitable in four cases were using Agile software development process, whereas one company which found ISO 15504 more suitable was using Waterfall software development process.

Keywords: Software Process Maturity, Software Quality, Software Process Improvement.

1 Giriş

Günümüzde organizasyonlar hızla değişen teknoloji ve beklentilere ayak uydurmak için sürekli bir değişim içerisindedir. Bu değişim, organizasyonun; stratejisinin, kültürünün, süreçlerinin, organizasyonun kendisi üzerinde oluşturduğu

etkileri kapsamaktadır. Yazılım organizasyonlarının, yenilikçi uygulamaları ve pratikleri değerlendirip, kendileri için uygun olanları seçerek değişmeleri yaşamsal öneme sahiptir. Bu amaç için 1990 yılından itibaren sektöre özgü yaklaşımlar geliştirilmiştir. Bu yaklaşımlar içerisinde en sık kullanılan SPICE (ISO–15504, ISO 3300X) [10,16] ve CMM (Capability Maturity Model) [17] olmuştur. Bu modellerin günümüz yazılım organizasyonlarında kullanımı ili ilgili belirgin fikir ayrılıkları ve tartışmalar olsa da Çevik (Agile) yazılım geliştirme süreçlerinin bu yaklaşımlarla, uygun bir şekilde iyileştirilebilirliği ve var olan yaklaşımlardan hangilerinin günümüz organizasyonlarına daha uygun olacağı ile ilgili kapsamlı bir çalışma bulunmamaktadır.

Bu araştırma kapsamında, önceden belirlenmiş organizasyonlar için, hem 15504 hem de AgilityMod yaklaşımları kullanılarak ayrı ayrı değerlendirmeler yapılmıştır. Bu değerlendirmeler ve sonuç analizleri; İzmir Yüksek Teknoloji Enstitüsü Bilgisayar Mühendisliği'nin CENG515 dersi kapsamında yüksek lisans ve doktora öğrencileri tarafından gerçekleştirilmiştir. Takımlar 5 ayrı metodolojiye sahip ve farklı büyüklükteki firmalar için her iki yaklaşımı da uygulayıp, hangi yaklaşımın, hangi özelliklere sahip organizasyonlar için daha uygun olacağını değerlendirmiş, sonuçları bu makalede tartışılmıştır.

Tüm değerlendirme ön sonuçları, diğer takımlar tarafından kritik edilip, gerekli ise revize ve geliştirme için tekrar ele alınmıştır. Tüm takımlar revizyonları tamamladıktan sonra, sonuçlar analiz edilmiş ve hangi organizasyonlar için hangi yaklaşımın daha uygun sonuçlar verdiğine dair incelemeler yapılmıştır. 2.bölümde; bahsedilen yaklaşımlarla ilişkili çalışmalar ve yaklaşımların özelliklerinin karşılaştırılması yapılmıştır. 3.bölümde; takımların gerçekleştirmiş oldukları durum çalışmalarının tasarımı ve durum çalışmalarının sonuçları detaylı bir şekilde anlatılmıştır. 4.bölümde; genel değerlendirme ve değerlendirme için kullanılan araçların analizi bulunmaktadır.

2 İlişkili Çalışmalar

Yazılım süreci olgunluğu değerlendirme ve iyileştirme modelleri birçok yazılım organizasyonu tarafından kullanılıp araştırmacılar tarafından incelenmiştir [7,8]. Bu modellerden en bilinenleri, Yetenek Olgunluk Modeli (CMM) [17], CMM'in ayrı modellerini bütünleştirmek ve standart hale getirmek amacıyla geliştirilen Yetenek Olgunluk Model Entegrasyonu (CMMI) [5] ve ISO 15504 / ISO 3300X Yazılım Süreç İyileştirme ve Yetenek Değerlendirme (SPICE) Modelidir [10,16]. Bu modeller farklı yazılım geliştirme süreç modelleri kullanan birçok organizasyon için değerlendirme kıstası olmuştur.

Müşterilerin ihtiyaçlarına daha çabuk cevap veren esnek bir yapıda olan değişime adapte olmayı kolaylaştıran çevik (Agile) yazılım geliştirme yaklaşımının [6] yaygın bir şekilde kullanılmaya başlanması ile diğer bilinen modelleri kullanan organizasyonların yanı sıra önemli sayıda çevik modeli kullanan organizasyonun yazılım pratiklerini değerlendirme ve geliştirme ihtiyacı ortaya çıkmıştır [9,19]. Bu alanda çalışma yapan araştırmacılar, çevik yaklaşımlar için ISO 15504 ve CMM ile değerlendirme ve iyileştirmenin yanında başka yaklaşımlara ihtiyaç

olduğunu belirlemişlerdir [15,12]. Bu ihtiyacı karşılamak amacıyla birçok araştırmacı çevik model için çok sayıda çevik olgunluk değerlendirme modeli geliştirilmiştir [11,14,21].

Humble ve Russell'ın Çevik Olgunluk Modeli [11] gibi çevik yazılım için geliştirilen modellerden bazıları kısa açıklamalar, öneriler veya anketler şeklinde iken, Patel ve Ramachandran'ın Çevik Olgunluk Modeli [14] ve Yin'in Scrum Olgunluk Modeli [21] gibi bazıları da detaylı bir analize izin veren daha kapsamlı yapıdadırlar. Bu modeller, amaçlarına yönelik birçok niteliği karşılarken, çevik olgunluk değerlendirmesinde belirgin eksik yönleri bulunmaktadır [23]. Patel ve Ramachandran'ın modeli, detaylı açıklamarın eksikliği sebebi ile bütünlük, kullandığı dil sebebi ile nesnellik bakımından eksik kalırken; Yin'in modeli sadece çevik yazılım geliştirme pratiği olan Scrum [18] için uygulanabilir yapıdadır ve genel bir çevik model kapsamı içermemektedir [13].

Özcan-Top ve Demirörs'ün AgilityMod çeviklik değerlendirme modeli [20], bahsi geçen modellere nazaran; kapsam, çeviklik seviyelerin uyumlu tanımı, nesnellik ve bütünlük nitelikleri ile çalışmamıza daha uygun bulunmuştur [22]. Ayrıca AgilityMod, ISO 15504 yaklaşımını temel alan bir notlandırma yaklaşımı içermektedir [20,22]. Bu sayede yapılan çalışmada ISO 15504 yaklaşımı ile yapılacak karşılaştırmayı kolaylaştırmakta ve geniş çapta bir değerlendirme kıstası sağlamaktadır. Bu sebeplerden ötürü, yaptığımız değerlendirme çalışmasında kullanılmasına karar verilmiştir.

ISO / IEC 15504, yazılım geliştirme süreçleri için değerlendirme kıstasları ile birlikte süreç geliştirme ve yetenek belirleme çerçevesidir [3]. 7 kısımdan oluşan 15504 modelinin 2. ve 5. kısımları AgilityMod için taban oluşturmuştur. İkinci bölümde değerlendirmenin gerçekleştirilmesi için gerekli aşamalar ve genel unsurlar tanımlıdır. Beşinci bölümde ise süreç değerlendirme modelinin detaylı bir tasviri bulunmaktadır. Yetenek ve süreç boyutları (capability dimension, process dimension), iş ürünleri ve uygulamalarıyla (work products and practices) tarif edilmiştir.

İki boyuttan oluşan 15504'ün yetenek boyutu, tamamlanmamıştan (incomplete) iyileştirilene (optimizing) kadar 6 seviyeden oluşur. Süreç boyutu ise 3 kategoride toplanan 48 farklı süreçten meydana gelir [1,2]. Bu 3 kategori ise kuruluşa ait (organizational), temel (primary) ve yardımcı (supporting) süreçler olarak tanımlanır. Altı yetenek seviyesi bu süreçlerden bağımsız olarak 9 farklı süreç niteliği (process attribute) ile incelenir.

AgilityMod, ISO / IEC 15504'ün ölçüm pratiğini temel alan çevik yazılım geliştirme pratikleri için tasarlanmış temel farklılıklar içeren bir çerçevedir [20]. AgilityMod, Bakış açısı ve Çeviklik boyutlarını içermektedir (aspect dimention, agility dimention). Çevik süreçlerden türetilmiş bakış açıları, keşif (exploration), geliştirme (construction), geçiş (transition) ve yönetim (management) olarak ayrılır. Diğer boyut olan çeviklik ise sırası ile uygulanmamış (not implemented), geçici (ad-hoc), yalın (lean) ve etkili (effective) olarak 4 seviyede incelenir. Başarısı elde edilen her bir seviye, ilgili bakış açısının çevikliğinin arttığının göstergesidir. Dört çeviklik seviyesi; basit (simple), tekrarlanan (it-

erative), teknik olarak mükemmel (technically excellent) ve öğrenen (learning) olarak 4 bakış açısı niteliği (aspect attribute) ile incelenir.

Her iki modelin de puanlama yöntemi aynıdır. İki modelde de uygulamalar, 4 seviyede derecelendirilir(Başarısız (Not Achieved), Kısmen Başarılı (Partially Achieved), Genellikle Başarılı (Largely Achieved), ve Tamamen Başarılı (Fully Achieved)).

3 Durum Çalışmaları

Durum çalışmaları için gereken verilerin toplanabilmesi için İzmir Yüksek Teknoloji Enstitüsü Bilgisayar Mühendisliği öğrencileri ile çalışılmıştır. CENG515 dersini alan, hepsi yüksek lisans ya da doktora yapan öğrenciler çalıştıkları yazılım geliştirme çalışmaları yürütülen kurumlarda bir proje ile ilgili hem 15504 hem de AgilityMod yaklaşımları kullanılarak ayrı ayrı değerlendirmeler yapmıştır. Toplamda 5 adet farklı büyüklükteki firmalar için her iki yaklaşım da değerlendirilmiş, hangi yaklaşımın, hangi özelliklere sahip organizasyonlar için daha uygun olacağı analiz edilmiştir.

Tablo 1'de, gerçekleştirilen durum çalışmaları ile ilgili genel bilgiler verilmektedir. Modellerin değerlendirme yapısından dolayı, değerlendirilen süreçlerde farklılıklar bulunmaktadır. Bu farklılıklar kurumun yazılım geliştirme yaşam döngüsünden kaynaklanmıştır.

Tablo 1. Gerçekleştirilen beş durum değerlendirmesi.

Durumlar	Şirket Sektörü	Şirket Boyutu	15504 Kapsamında Değerlendirilen Süreç	AgilityMod Kapsamında Değerlendirilen Süreç
Durum A	Bankacılık	450 kişilik yazılım ekibi	ENG.8 - Yazılım Test Süreci	Yazılım Geliştirme Süreci
Durum B	Ulaşım	35 kişilik yazılım ve donanım ekibi	SPL.2 - Ürünün piyasaya sürümü	Değişim Süreci
Durum C	Güvenlik Yazılımları	11 kişilik yazılım ekibi	ENG.6 - Yazılım Geliştirme (Software Construction)	Yazılım Geliştirme Süreci
Durum D	Bilişim	11 kişilik yazılım ekibi	ENG.11 - Yazılım Bakım Süreci	Yazılım Geçiş Süreci
Durum E	Bankacılık	450 kişilik yazılım ekibi	ENG.6 - Yazılım Geliştirme (Software Construction)	Yazılım Geliştirme Süreci

Durum A için ele alınan şirket, Scrum yöntemi ile yazılım geliştirmektedir. Yazılım geliştirme çalışmaları 2-3 haftalık döngülere (sprint) bölünmüştür. Her döngü başında planlama, her döngü sonunda ise yeniden inceleme (review) ve süreç değerlendirme toplantıları (retrospective meetings), ve her sabah günlük toplantılar gerçekleştirilmektedir. Bu kurumda 15504 ile seviye 3'e kadar yazılım test süreci değerlendirilmiştir. Değerlendirmeler yapılırken şirketin mevcut yapısı ve bu yapıyı nasıl geliştirebileceği konuları da ele alınmıştır. Değerlendirmeler "ISO 15504-5 An Exemplar Process Assessment Model" dokümanı referans alınarak yapılmıştır. Değerlendirme 4 kişilik bir ekip ile 2 kişi/gün sürmüştür. Değerlendirmenin sonuçları Tablo 2'de verilmiştir .

AgilityMod modeliyle yapılan değerlendirme sırasında Yapım Bakış Açısı değerlendirilmiştir. Değerlendirmeler gerçekleştirilirken bir yazılım aracı (AssessAgility) kullanılmıştır [4]. Bu araç değerlendirme sürecinde ve değerlendirme sonucunda raporlama kısmında oldukça yararlı olmuştur. Değerlendirme 4 kişilik bir ekip için 1 kişi-gün sürmüştür. Sürenin kısalmasındaki önemli etkenler

Tablo 2. Durum A 15504 Değerlendirme Sonuçları.

Attributes	PA 1.1 Process Performance	PA 2.1 Performance Management	PA 2.2 Work Product Management	PA 3.1 Process Definition	PA 3.2 Process Deployment
Durum A Sonuçları	Fully Achieved	Fully Achieved	Fully Achieved	Fully Achieved	Fully Achieved

kullanılan araç ve takımın deneyimi olmuştur. AgilityMod modeliyle yapılan değerlendirme sonucunda ise şirketin seviyesi "seviye 2 Lean" olarak bulunmuştur. Bu seviye kullanılan aracın raporunda aşağıdaki Tablo 5'deki gibi gösterilmiştir.

Tablo 3. Durum A AgilityMod Değerlendirme Sonuçları.

	1.Ad-Hoc				2.Lean				3.Effective					
	AP1	AP2	AP3	AP4	GP 2.1.1	GP 2.1.2	GP 2.2.1	GP 2.2.2	GP 3.1.1	GP 3.1.2	GP 3.2.1	GP 3.2.2	GP 3.2.3	GP 3.2.4
Durum A	FA	FA	FA	FA	FA	FA	LA	FA	FA	FA	FA	PA	FA	FA

Modellerde değerlendirilmek üzere A şirketinden "Bilgi Teknolojileri Proje Yönetim" yaşam döngüsü seçilmiştir. 15504 için test aşaması, AgilityMod için ise geliştirme aşaması ele alınmıştır. Ek olarak, bilindiği üzere Scrum yöntemi ile geliştirilen projelerde , analiz, tasarım, geliştirme ve test süreçleri birbirini takip eden şekilde ve iç içe ilerlemektedir. Bu nedenle A şirketinde seçilen süreci sadece test boyutundan ele almak zor ve kısıtlayıcı olmuş ve şirkette gerçekleştirilen sürece tam uymayan bir değerlendirmeye yol açmıştır. AgilityMod modeli ise ilgili bakış açısının bütün olarak değerlendirilmesine olanak sağlamıştır. Çünkü AgilityMod modelinin yapısı Agile olarak geliştirilen pratiklere uygun olarak, değerlendirmeye uyumlu sınırlar getirmektedir. Süreçler değerlendirilirken bütün olarak değerlendirilebilmiştir. Ve bu değerlendirmenin A şirketi açısından gerçekleştirilen etkinliklere daha yakın olduğu gözlenmiştir.

Durum B için ele alınan şirket, Şelale (Waterfall) yöntemi ile yazılım geliştirmektedir. Müşteri ile yapılan anlaşmalar sonucu; analiz, tasarım, kodlama, test ve entegrasyon adımları sırası ile işlemektedir. Müşterilerin ilgili adımlarda içeriği gözlemleme ve müdahale yetkisi, gizlilik sebebi ile bulunmamaktadır. 15504 ile seviye 3'e kadar değerlendirme yapılmıştır. Değerlendirmelerde şirketteki tedarik yapısı ele alınmıştır. Değerlendirmeler "ISO 15504-5 An Exemplar Process Assessment Model" dokümanı referans alınarak, bir araç kullanılmadan yapılmıştır. Değerlendirme 2 kişilik bir ekip tarafından gerçekleştirilmiş, 2 kişi/gün iş gücü gerektirmiştir. Değerlendirmenin sonuçları Tablo 4'de verilmiştir .

Tablo 4. Durum B 15504 Değerlendirme Sonuçları.

Attributes	PA 1.1 Process Performance	PA 2.1 Performance Management	PA 2.2 Work Product Management	PA 3.1 Process Definition	PA 3.2 Process Deployment
Durum B Sonuçları	Fully Achieved	Fully Achieved	Largely Achieved	Largely Achieved	Partially Achieved

AgilityMod modeliyle yapılan değerlendirme Geçiş (Transition) bakış açısından yapılmıştır. Değerlendirmeler gerçekleştirilirken bir araç kullanılmamıştır. Değerlendirme 2 kişilik bir ekip tarafından yapılmış 2 kişi-gün efor gerektirmiş-

tir. AgilityMod modeliyle yapılan değerlendirme sonucunda ise şirketin seviyesi "seviye 1 Ad-Hoc" olarak bulunmuştur. Değerlendirmenin sonucu aşağıdadır.

Tablo 5. Durum B AgilityMod Değerlendirme Sonuçları.

	1.Ad-Hoc						2.Lean				3.Effective					
	AP1	AP2	AP3	AP4	AP5	AP6	GP 2.1.1	GP 2.1.2	GP 2.2.1	GP 2.2.2	GP 3.1.1	GP 3.1.2	GP 3.2.1	GP 3.2.2	GP 3.2.3	GP 3.2.4
Durum B	FA	LA	FA	PA	NA	NA	NA	NA	FA	PA	NA	PA	FA	NA	FA	NA

Modellerde değerlendirilmek üzere B şirketinden seçilen "Tedarik ve Bakım" süreci 15504 ile birebir uyumludur. Şelale yöntemi ile ilerleyen süreçlere sahip B şirketi için 15504 değerlendirmesi daha kolay olmuştur. Şelale modeli ile uyumlu bir süreç mimarisine sahip olan 15504 ile şirket için seçilen sürece karşılık gelen uyumlu süreç kolayca bulunmuştur. AgilityMod modeli ile değerlendirmede eksikliklere sebep olmuştur. Çevik modellere uygun şekilde tasarlanan AgilityMod, şelale yöntemi ile ilerleyen şirketin tedarik ve bakım sürecine birebir karşılık gelen bir bakış açısını içermemektedir.

Durum B için gerçekleştirilen karşılaştırma sonucunda 15504 daha yararlı bulunmuştur. Bunun sebebi Durum B içinde değerlendirilen sürecin şelale yöntemi ile ilerlemesi ve her bir sürecin birbirinden bağımsız olmasıdır. AgilityMod içerisinde değerlendirilen sürece karşılık uygun süreç tam olarak bulunamamıştır ve bulunan bakış açıları süreç bazında değerlendirmeye uygun değildir. Çeviklik boyutundan yapılan inceleme süreç için gerekli iyileştirme önerilerini bulmada yetersiz kalmıştır. Bu durum değerlendirmenin uzamasına sebep olmuştur. Bu noktada 15504, AgilityMod'a göre iyileştirme önerilerinin daha iyi görülmesini sağlamıştır. Çeviklik boyutundan yapılan değerlendirme, şelale yöntemi ile ilerleyen süreçler için anlamlı sonuçlar üretmede kısır kalırken; olgunluk boyutundan yapılan değerlendirme, faydalı iyileştirme önerilerinin bulunmasını sağlamıştır. Sonuç olarak 15504 ile yapılan değerlendirme, AgilityMod ile yapılan değerlendirmeye göre kısa sürmüş ve daha yararlı sonuçlar vermiştir.

Durum C'de ele alınan şirket, 11 çalışana sahip ve güvenlik yazılımlar üreten bir girişim şirketidir. Yazılım geliştirmede Scrum yöntemi kullanılmakta ve her bir döngü (sprint) genel olarak 2 hafta sürmektedir. Süreç yeteneği değerlendirmesinde ISO 15504 Spice modelinde ENG 6. Yazılım Geliştirme (Software Construction) seviye 3, AgilityMod modelinde ise Geliştirme (Construction) Bakış Açısı ele alınmıştır. ISO 15504 Spice değerlendirmesi 3 kişilik bir ekip tarafından 2 kişi-günde yapılmıştır. Değerlendirme sonuçları Tablo 6 'de verilmiştir.Değerlendirme sonucuna göre şirket Yetenek seviyesi 1'i tam olarak başaramamış durumdadır.

Tablo 6. Durum C 15504 Değerlendirme Sonuçları.

Attributes	PA 1.1 Process Performance	PA 2.1 Performance Management	PA 2.2 Work Product Management	PA 3.1 Process Definition	PA 3.2 Process Deployment
Durum C Sonuçları	Partially Achieved	Largely Achieved	Partially Achieved	Partially Achieved	Partially Achieved

AgilitMod modelinde değerlendirme 3 kişilik bir ekip tarafından 3 kişi-günde gerçekleştirilmiştir. Değerlendirme sonucuna göre şirket Yetenek seviyesi 1'i tam olarak başaramamış durumdadır.

Tablo 7. Durum C AgilityMod Değerlendirme Sonuçları.

	1.Ad-Hoc				2.Lean				3.Effective					
	AP1	AP2	AP3	AP4	GP 2.1.1	GP 2.1.2	GP 2.2.1	GP 2.2.2	GP 3.1.1	GP 3.1.2	GP 3.2.1	GP 3.2.2	GP 3.2.3	GP 3.2.4
Durum C	LA	LA	FA	LA	FA	LA	LA	LA	PA	LA	FA	FA	PA	PA

ISO 15504 modelinde tanımlanan süreç sadece yazılım geliştirme sürecini ele almakta, AgilityMod modelinde ise ele alınan Yapım (Construction) Bakış Açısı ise yazılım geliştirme sürecini test süreci ile birlikte ele almaktadır. C şirketinde Agile yazılım geliştirmeye yönelik bir süreç izlendiği için ISO 15504 modelindeki Yazılım Geliştirme süreci tek başına değerlendirildiğinde Agile yazılım geliştirme yapısına tam uymamış ve değerlendirmeyi karmaşıklaştırmıştır. Yazılım Geliştirme Süreçi çıktılarının karşılıklarını bulmak zor olmuştur. AgilityMod modelinde ise yazılım geliştirme ve test süreçleri bir bütün olarak ele alındığından C şirketinde yapılan değerlendirme daha kolay olmuş ve yürütülen etkinlikler ile uyumlu olmuştur. Sonuç olarak, AgilityMod modeli ile değerlendirme C şirketinde uygulanan sürece daha uygun olmuştur.

Durum C için genel olarak ele aldığımızda, AgilityMod ile değerlendirme daha kullanışlı ve sonuçlar daha yararlı bulunmuştur. C şirketinde çevik yazılım geliştirme mantığı benimsenmiş olduğu için ISO 15504 Spice Yazılım Geliştirme süreci dokümantasyon ve tanımlanması gereken süreçler yönünden şirkette pek karşılığını bulamamıştır. Bu nedenle ISO 15504 Spice modeli süreçlerin yanı sıra olgunluk (maturity) boyutunda da ele alınan şirket için uygulanabilir önerilerde bulunamamıştır. İki modeli karşılaştırdığımızda ise yine AgilitMod'un çevik yazılım geliştirmeye daha uygun olduğu görülmektedir. Değerlendirme sürecinde AgiltyMod'un şirkette var olan sürece daha uygun olduğu gözlenlenmiştir. Değerlendirme sonucu yapılan süreç iyileştirme önerilerde AgilitMod yine şirketteki yapıya uygunluğundan dolayı daha yararlı görülmüştür.

Durum D için ele alınan şirket, Scrum yöntemi kullanarak yazılım geliştirmektedir. Haftalık döngüler (sprints) ile yazılım geliştirme süreci parçalara ayrılır. Her döngü başında, genel kaynak planlaması, her gün sonunda yarım saat olarak belirlenen, günlük gelişmelerin gözden geçirilmesi ve 2 günde bir olmak üzere her sabah yine yarım saatlik günlük değerlendirme toplantıları yapılır. 15504 ile seviye 3'e kadar değerlendirme yapılmıştır. Değerlendirme ISO 15504-5 modeli referans alınarak şirketin mevcut yapısında bulunan eksiklikleri tanımlamak üzere, manuel olarak gerçekleştirilmiştir. Değerlendirme 3 kişilik bir ekip için 3 kişi-gün sürmüştür. Değerlendirmenin sonuçları Tablo 8'de verilmiştir.

AgilityMod modeliyle yapılan değerlendirme sırasında Geçiş (Transition) Bakış Açısı incelenmiştir. Değerlendirmeler, modelin ilgili aracı kullanılarak yapılmış (AgilityMod) ve değerlendirme sonucu raporlama/dokümantasyon olmak üzere araç sayesinde otomatik elde edilmiştir. Değerlendirme, 3 kişi-gün sürmüş-

Tablo 8. Durum D 15504 Değerlendirme Sonuçları.

Attributes	PA 1.1 Process Performance	PA 2.1 Performance Management	PA 2.2 Work Product Management	PA 3.1 Process Definition	PA 3.2 Process Deployment
Durum D Sonuçları	Largely Achieved	Largely Achieved	Largely Achieved	Largely Achieved	Largely Achieved

tür. Bu sonuca göre test ettiğimiz şirketin seviyesi "seviye 2 Lean" olarak tespit edilmiştir. Bu seviye için oluşturulan rapor, Tablo 9'de verilmiştir.

Tablo 9. Durum D AgilityMod Değerlendirme Sonuçları.

	1.Ad-Hoc				2.Lean				3.Effective					
	AP1	AP2	AP3	AP4	GP 2.1.1	GP 2.1.2	GP 2.2.1	GP 2.2.2	GP 3.1.1	GP 3.1.2	GP 3.2.1	GP 3.2.2	GP 3.2.3	GP 3.2.4
Durum D	FA	FA	FA	FA	FA	FA	LA	FA	FA	FA	FA	PA	FA	FA

Modellerde değerlendirmek üzere D şirketinden "Private Historical Dashboarding Software" projesi seçilmiştir. 15504 için bu projenin bakım ve onarım süreci, AgilityMod için ise bu projenin geçiş ve geliştirme bakış açısı ele alınmıştır. Ek olarak, D şirketinde bakım ve onarım işlemleri, müşteriden gelen geri bildirimler göz önünde bulundurularak, acil çözüm ekipleri ile müşteri odaklı çözüme ulaştırılır. Bu da gösteriyor ki, D şirketinde geliştirme süreci, proje müşteriye ulaştıktan sonra dahi devam eder. Bu sebeple, ISO 15504-5 modelinde bize en uygun olarak belirlenen ENG.11 Yazılım Bakımı (Software Maintenance) süreci, sorunların ve çözümlerin önceden belirlenmesi gibi şirketin geliştirme sürecine uygun olmayan gerekliliklerinden ötürü, D şirketinin belirlenen sürecini tam anlamıyla analiz etmesi açısından uyumlu değildir.

AgilityMod modeli ise ISO 15504'e nazaran daha başarılı bir profil oluşturmuştur. Esnek yapısı ve çevik proje geliştirme prensipleri göz önüne alınarak geliştirilmesi, D şirketinde belirlenen kapsamın analizini daha kolay ve daha gerçeğe yakın yapabilmemize olanak sağlamıştır.

Durum D için ISO 15504 ve AgilityMod karşılaştırması sonucunda AgilityMod yaklaşımının daha faydalı sonuçlar ürettiği değerlendirilmiştir. Buna ek olarak, seçilen süreç tanımı ISO 15504 model ile tam olarak örtüşse de Scrum yönetimi için anlamlı sonuçlar vermemiştir. AgilityMod süreçleri daha genel bir tanım olarak sınıflandırmasına karşın şirket yapısı için daha hızlı ve kullanışlı sonuçlar vermiştir. Agility mod için kullanılan araç modellemenin oluşmasında kolaylık sağlamıştır.

Durum E için ele alınan şirket, durum A'da ele alınan şirket ile aynı olup, 15504 kapsamında farklı bir proje değerlendirilmiştir. Süreç yeteneği değerlendirmesinde ISO 15504 Spice modelinde ENG 6. Yazılım Geliştirme (Software Construction) seviye 3, AgilityMod modelinde ise Geliştirme Bakış Açısı (Construction Aspect) ele alınmıştır. ISO 15504 Spice değerlendirmesi 3 kişi-günde yapılmıştır. Değerlendirme profili Tablo 10 'de özetlenmiştir.

AgilitMod modelinde değerlendirme 3 kişi-günlük bir efor gerektirmiştir. Değerlendirme sonucuna göre şirket Yetenek seviyesi 3'e ulaşmış durumdadır.

Tablo 10. Durum E 15504 Değerlendirme Sonuçları.

Attributes	PA 1.1 Process Performance	PA 2.1 Performance Management	PA 2.2 Work Product Management	PA 3.1 Process Definition	PA 3.2 Process Deployment
Durum E Sonuçları	Fully Achieved	Fully Achieved	Fully Achieved	Partially Achieved	Partially Achieved

Tablo 11. Durum E AgilityMod Değerlendirme Sonuçları.

	1.Ad-Hoc				2.Lean				3.Effective					
	AP1	AP2	AP3	AP4	GP 2.1.1	GP 2.1.2	GP 2.2.1	GP 2.2.2	GP 3.1.1	GP 3.1.2	GP 3.2.1	GP 3.2.2	GP 3.2.3	GP 3.2.4
Durum E	FA	FA	FA	FA	LA	FA	FA	LA	FA	LA	LA	FA	FA	FA

Durum E için ISO 15504 ve AgilityMod karşılaştırması sonucunda AgilityMod'un şirket yapısı ile daha uyumlu olduğu sonucuna varılmıştır. Buna ek olarak, seçilen kapsam ISO 15504 model ile örtüşse de uygulanan etkinlikler için yararlı sonuçlar vermemiştir. Olgunluk boyutundan yapılan değerlendirme, uygulanabilir iyileştirme önerilerinin bulunmasında başarısız olmuştur. E şirketinde çevik yazılım geliştirme mantığı benimsenmiş olduğu için ISO 15504 Spice Yazılım Geliştirme süreci dokümantasyon ve tanımlanması gereken süreçler yönünden şirkette pek karşılığını bulamamıştır. AgilityMod süreçleri daha genel bir tanım olarak sınıflandırmasına karşın şirket yapısı için daha uyumlu ve kullanışlı sonuçlar üretmiştir. Çeviklik boyutundan yapılan değerlendirme, şirket için faydalı iyileştirme önerilerinin bulunmasını sağlamıştır.

4 Sonuç

Bu çalışmada, İzmir Yüksek Teknoloji Enstitüsü Bilgisayar Mühendisliği CENG 515 dersini alan yüksek lisans ve doktora öğrencileri tarafından 5 ayrı durum ele alınarak farklı şirketlerde SPICE Süreç Yeteneği Değerlendirme ve AgilityMod Çeviklik Yeteneği Değerlendirme modellerine göre değerlendirme yapılmıştır. Değerlendirmeler, Bankacılık, Ulaşım, Güvenlik Yazılımı, Bilişim sektörlerinde, farklı büyüklüklerde 4 ayrı şirkette (2 grup aynı şirketin farklı projeleri ve süreçlerini ele almıştır) farklı süreçler ve bakış açıları üzerinde yapılmıştır. Şirketlerde uygulanan süreçlerin bulundukları yetenek seviyeleri dersi alan öğrenci grupları tarafından ISO 15504 SPICE süreçlerine ve ilgili AgilityMod bakış açılarına göre analiz edilmiştir. ISO 15504 SPICE modeline göre yapılan değerlendirmede, ele alınan durumlardan biri seviye 3, biri seviye 2, ikisi seviye 1 düzeyinde, biri ise seviye 1'in altında çıkmıştır. AgilityMod'a göre yapılan değerlendirmede ise biri seviye 3, ikisi seviye 2, biri seviye 1 düzeyinde, biri ise seviye 1'in altında çıkmıştır. Ele alınan 3 ayrı durumda ISO 15504 SPICE ve AgilityMod'da farklı sonuçlar, diğer 2 durumda ise benzer yetenek seviyesi sonuçları elde edilmiş gibi görünse de AgilityMod ve SPICE yetenek seviyeleri birbirinden farklı olduğundan karşılaştırılabilir değildir.

Yetenek seviyesi değerlendirmeleri yapıldıktan sonra gruplardan bu iki model ile ilgili karşılaştırma yapılması istenmiştir. Bu karşılaştırmada hangi modelin ele aldıkları şirket için daha uygun olduğu, modellerde seçilen süreçlerin şirketlerde işletilen etkinlikler ile uygunluğu ve elde edilen sonuçların şirkete olan yararı değerlendirilmiştir. Elde edilen sonuçlara göre, 4 durum için AgilityMod'un bakış

açıları ve yetenek seviyeleri açılarından uygunluğu değerlendirilmiştir. Karşılaştırmalarda genel olarak ISO 15504 SPICE süreçlerinin şirket yazılım geliştirme yapısına tam örtüşmediği ve bu süreçler üzerinden yapılan değerlendirmede zorlanıldığı, AgilityMod bakış açılarının şirket yazılım geliştirme yapısına daha uygun olduğu ve sonuçların şirkette uygulanabilir olduğu belirtilmiştir. AgilityMod'u daha uygun bulan 4 durumda incelenen şirketler Çevik yazılım geliştirme yaklaşımı, uygun bulunmayan şirkette ise Şelale yazılım geliştirme yaklaşımı kullanılmaktadır. Uygunluğa ek olarak, AgilityMod değerlendirmesi çoğu grupta daha kısa sürede tamamlanmıştır. AgilityMod değerlendirmelerinin girildiği ve sonuç raporunun otomatik olarak üretildiği bir araç da ayrıca kullanılmıştır. Bu aracın değerlendirmeyi kolaylaştırdığı belirtilmiştir.

ISO 15504 SPICE modeli süreçleri Şelale yaşam döngüsü ile uyumlu bir mimaride tanımlamaktadır. Bu yapı Şelale yaşam döngüsü ile yazılım geliştiren şirketler için daha uygun olmaktadır. Çevik prensiplerle uyumlu yazılım geliştiren şirketlerde ise yazılım bu bu tanımlı yöntemlerin belirlediği pratiklere bölünmekte bu da ISO 15504 SPICE modeli ile değerlendirilecek sürecin sınırlarını belirlemede güçlük yaratmaktadır. Ayrıca ISO 15504 SPICE modelinde belirtilen süreçler çevik yazılım geliştirme pratiklerine tam örtüşmediğinden denetlemede bu pratiklerin şirkette tam anlamıyla uygulandığının belirlenmesi güç olmaktadır. AgilityMod modeli çevik yazılım geliştirme pratiklerini temel alarak geliştirildiğinden; çevik yazılım geliştiren şirketlerin uygulamalarının değerlendirilmesine daha uygun olmaktadır. AgilityMod modelindeki bakış açısı yaklaşımı çevik yazılım geliştirmedeki gibi bu pratikleri bir bütün olarak ele almakta ve çevik yazılım geliştirme pratiklerinin uygulanabilirliğini sorgulamaktadır. Bu yapı çevik yazılım geliştiren şirketlerde pratiklerin tam anlamıyla uygulandığını belirlemede ve sonuçlara göre iyileştirme yapılmasında kolaylık sağlamaktadır.

ISO 15504 SPICE modeli yazılım geliştirme süreçleri dışında farklı süreç gruplarını da içermektedir. Bu yazılım geliştirme süreçleri dışında faklı süreçlerin yeteneğini belirlemek isteyen şirketlere yararlı olmaktadır. AgilityMod ise yazılım geliştirme süreçlerini kapsadığı için bu süreçler dışında farklı süreçlerin yeteneğinin belirlenmesinde kullanılamamaktadır.

ISO 15504 SPICE modelinde yer alan yetenek boyutu çevik yazılım geliştiren 4 şirket için tam anlamıyla karşılık bulamamıştır. Çevik yazılım geliştiren şirketlerde basamaklı bir yapı olmadığından yetenek boyutunun verdiği iyileştirme önerileri bu şirketler için uyumlu olmamıştır. AgilitMod modelindeki çeviklik boyutu ise bu 4 şirkette daha uygun iyileştirme önerileri oluşturmuştur.

Kaynakça

1. ISO/IEC 12207:1995/Amd.1:2002. Information technology — Software life cycle processes (2002)
2. ISO/IEC 12207:1995/Amd.2:2004 (2004)
3. International Organization for Standardization. ISO/IEC 15504 International Standard Information Technology – Software Process Assessment: Part 1–Part 7 (2008)

4. Adali, O.E., Top, Ö.Ö., Demirörs, O.: Assessment of agility in software organizations with a web-based agility assessment tool. In: 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA). pp. 88–95. IEEE (2017)
5. Ahern, D.M., Clouse, A., Turner, R.: CMMI distilled: a practical introduction to integrated process improvement. Addison-Wesley Professional (2004)
6. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al.: Manifesto for agile software development (2001)
7. Becker, J., Knackstedt, R., Pöppelbuß, J.: Developing maturity models for it management. Business & Information Systems Engineering **1**(3), 213–222 (2009)
8. Becker, J., Niehaves, B., Poeppelbuss, J., Simons, A.: Maturity models in is research. (01 2010)
9. Elssamadisy, A.: Agile Adoption Patterns: A Roadmap to Organizational Success (Adobe ebook). Addison-Wesley Professional (2008)
10. Garcia, S.M.: Evolving improvement paradigms: capability maturity models and iso/iec 15504 (pdtr). Software Process: Improvement and Practice **3**(1), 47–58 (1997)
11. Humble, J., Russell, R.: The agile maturity model applied to building and releasing software. ThoughtWorks White Paper (2009)
12. Łukasiewicz, K., Miler, J.: Improving agility and discipline of software development with the scrum and cmmi. IET software **6**(5), 416–422 (2012)
13. Ozcan-Top, O., Demirörs, O.: Assessment of agile maturity models: a multiple case study. In: International Conference on Software Process Improvement and Capability Determination. pp. 130–141. Springer (2013)
14. Patel, C., Ramachandran, M.: Agile maturity model (amm): A software process improvement framework for agile software development practices. International Journal of Software Engineering, IJSE **2**(1), 3–28 (2009)
15. Paulk, M.C.: Extreme programming from a cmm perspective. IEEE software **18**(6), 19–26 (2001)
16. Paulk, M.C.: A history of the capability maturity model for software. ASQ Software Quality Professional **12**(1), 5–19 (2009)
17. Paulk, M.C., Curtis, B., Chrissis, M.B., Weber, C.V.: Capability maturity model, version 1.1. IEEE software **10**(4), 18–27 (1993)
18. Schwaber, K.: Agile project management with Scrum. Microsoft press (2004)
19. Smith, G., Sidky, A.: Becoming Agile:–in an Imperfect World. Manning Greenwich (2009)
20. Top, O., Demirörs, O.: Agilitymod: Software agility assessment reference model v3. 0 application: Case study results. In: Technical Report METU/II-TR-2014–40. Informatics Institute. Middle East Technical Universit (2014)
21. Yin, A., Figueiredo, S., da Silva, M.M.: Scrum maturity model. Proceedings of the ICSEA pp. 20–29 (2011)
22. Yürüm, O.R., Demirörs, O.: Agile maturity self-assessment surveys: a case study. In: 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA). pp. 392–399. IEEE (2017)
23. Yürüm, O.R., Demirörs, O., Rabhi, F.: A comprehensive evaluation of agile maturity self-assessment surveys. In: International Conference on Software Process Improvement and Capability Determination. pp. 300–315. Springer (2018)

Ar- Ge Merkezleri için ISO/IEC 15504 SPICE Temelli Süreç Modelinin ve Yönetim Yazılımının Geliştirilmesi

Cihan Kabran[1], Erdem Özgür[1], Engin Hazar[1] ve Nihal Bilal[1]

[1]Teracity Yazılım Teknolojileri Ltd.Şti., Bursa, Türkiye
{cihan.kabran, erdem.ozgur, engin.hazar, nihal.bilal}@teracity.com.tr

Özet. Bu çalışmada ülkemizde özellikle yazılım konusunda faaliyet gösteren Ar-Ge merkezlerinin kurumsal süreçlerinin yönetimi ile proje yönetimi, destek ve yazılım mühendisliği süreçlerinin yönetimi için yapılmış bir süreç iyileştirme ve belgelendirme çalışması sunulmuştur. Yapılan çalışmada Bilim Sanayi ve Teknoloji Bakanlığı'nın Ar-Ge merkezlerinden beklentileri temel alınarak geliştirilen bir yönetim sistemi yazılımı üzerine Yazılım Süreç Olgunluğu modeli olarak tanımlanan ISO/IEC 15504 TR SPICE gereksinimleri göz önüne alınarak yapılan geliştirmeler ve sağladığı katkılar anlatılmıştır. Yazılım altyapısı sayesinde üretilen dokümanların sayısı ve yapılacak işler azaltılmakta, işlerin takibi kolaylaşmaktadır. Bu altyapının kuruma sağladığı en önemli katkı SPICE süreçlerinin sürdürülebilir hale gelmesidir. İş ve projelerin yönetimde kullanılan yazılımda, bütün veriler veri tabanında sürekli saklanmaktadır. Bundan dolayı SPICE süreç performans hedeflerinin bu verilere göre tanımlanması, performans hedeflerinin belirlenmesi ve hedeflerden sapmalarda gerekli önleyici faaliyetlerin düzenlenmesi kolaylıkla sağlanmaktadır. Kurumdaki çalışma süreçlerinin düzenli olarak izlenebilmesi doğal ve sürdürülebilir bir iyileşmeyi de sağlanmaktadır.

Anahtar Kelimeler: ISO/IEC 15504, SPICE, Süreç İyileştirme.

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Development of ISO / IEC 15504 SPICE Based Process Model and Management Software for R & D Centers

Cihan Kabran[1], Erdem Özgür[1], Engin Hazar[1] ve Nihal Bilal[1]

[1]Teracity Yazılım Teknolojileri Ltd.Şti., Bursa, Turkey
{cihan.kabran, erdem.ozgur, engin.hazar, nihal.bilal}@teracity.com.tr

Abstract.In this study, a process improvement and certification study for the management of corporate processes of R & D centers operating in the field of software and project management, support and software engineering processes in our country is presented. In this study, the developments and contributions made by considering ISO / IEC 15504 TR SPICE requirements defined as Software Process Maturity model on a management system software developed based on the expectations of the Ministry of Science, Industry and Technology from R & D centers are explained. Thanks to the software infrastructure, the number of documents produced and the work to be done are reduced and the follow-up of the works is easier. The most important contribution of this infrastructure to the organization is the sustainability of SPICE processes. In the software used in the management of business and projects, all data is stored in the database continuously. Therefore, it is easily provided to define SPICE process performance targets according to these data, to determine performance targets and to organize necessary preventive actions in deviations from the targets. Regular monitoring of the work processes in the organization ensures a natural and sustainable improvement.

Keywords: ISO/IEC 15504, SPICE, Process Improvement.

1 Giriş

Yazılım projelerinin zamanında, bütçesinde, belirlenen kapsamında ve hedeflenen kalite gereklerini tutturabilmesi için, kurum yönetim ve teknik süreçlerinin belli bir olgunlukta olması ve özellikle proje yönetim, destek ve mühendislik süreçlerinin üst düzeyde bulunması gerekmektedir. Bir yazılım firmasının yönetim ve iş süreçlerinin, bir süreç iyileştirme modeli kullanılarak geliştirilmesi, olgunluk durumunun derecelendirilmesi ve bir sertifika ile belgelendirilmesi dünyada ve ülkemizde oldukça yaygınlaşmıştır. Günümüzde en çok CMMI [1] veya çoğunlukla sadece SPICE (Software Process Improvement and Capability Determination) Yazılım Süreç İyileştirme ve Olgunluk Belirleme modeli olarak adlandırılan, ISO/IEC 15504 TR [2, 3] modeller, kurum süreçlerinin iyileştirmesinde ve olgunluk durumunun derecelendirilmesinde kullanılmaktadır.

1986 yılında kurulan Teracity Yazılım Haziran 2015'ten günümüze Bilim Teknoloji ve Sanayi Bakanlığı tarafından onaylı Ar-Ge Merkezi olarak faaliyetlerine devam etmektedir. Şirketimiz yönetim bilgi sistemleri, yapay zeka ve süreç yönetimi ağırlıklı yazılım geliştirmektedir. Bu çalışmada SPICE modelinin, Teracity'de yaklaşık 3 yıldır kullanılan, firmanın kendi geliştirdiği ve Bilimp adı ile aynı zamanda satışa sunduğu Yönetim Bilgi Sistemi yazılımına nasıl entegre edildiği, kurumdaki Proje Yönetim, Destek ve Yazılım Mühendisliği süreçlerinin nasıl hızlı ve kolay bir şekilde iyileştirildiği, bunun sonucunda da sadece belgelendirme amaçlı yapılmış bir çalışma halinden çıkartılarak bir kültür haline getirdiği anlatılmaktadır.

Bildirinin genel yapısı şu şekildedir: Bölüm 2'de ISO/IEC TR 15504 SPICE Süreçleri Ölçüm Çerçevesinin en önemli kısımları kısaca özetlenmektedir. Bölüm 3'te bu çalışmada geliştirilen Yönetim Bilgi Sistemi ile SPICE Süreçlerinin Yönetiminin nasıl yapıldığı anlatılmaktadır. Bölüm 4'te, süreç performans hedeflerinin tanımlanması ve ölçümleri sunulmaktadır. Son bölüm olan Bölüm 5'de, Proje Yönetimi, Destek ve Yazılım Mühendisliği süreçleri iyileştirme çalışmalarında elde edilen önemli bilgi ve tecrübeler sunulmaktadır ve son bir değerlendirme yapılmaktadır.

2 SPICE Ölçüm Çerçevesi

ISO/IEC TR 15504 SPICE standartı yazılım geliştirme süreçleri ve ilgili iş yönetim fonksiyonları için yapılandırılmış süreçleri iyileştirme ve değerlendirme çerçevesi (assessment framework) sunmaktadır [2-3].

Olgunluk boyutu ve süreç boyutu olarak tarif ettiğimiz bu çerçevede olgunluk boyutu, Eksik" (Incomplete) seviyesinden (Seviye 0) "En iyileştirmeye" (Optimizing) (Seviye 5) kadar her bir sürecin bağımsız olarak ölçülebileceği, olgunluk seviyelerini içerir. Her seviye, süreç nitelikleri (process attributes) ile tanımlanır. Diğer yandan süreç boyutu, ISO/IEC 12207 - "Sistem ve Yazılım Mühendisliği - Yazılım Yaşam Döngüsü Süreçleri" ne uygun olan tanımlandığı süreç gruplarını içerir [4]. Teracity gibi sadece yazılım geliştiren bir yazılım firması için, başlangıç düzeyi olan SPICE

Seviye 2 yani "Performans ve İş Ürünü Yönetimi "belgesinin alınabilmesi için, Proje Yönetim, Destek ve Mühendislik alanlarından aşağıda Şekil 1'de görülen en az 15 sürecin yerine getirilmesi gerekmektedir. Bu 15 süreç Teracity'nin bu çalışmada anlatılan süreç iyileştirme kapsamını belirlemiş olmaktadır. Bildirinin bir sonraki bölümünde bu 15 süreç için yapılan Proje Yönetim, Destek ve Mühendislik çalışmaları ile bunların geliştirilen Yönetim Bilgi Sistemiyle nasıl bütünleştirildiği sunulmaktadır.

3 SPICE Süreçlerinin Yönetimi

BİLİMP Yönetim Bilgi Sistemi [5] yazılımı, öncelikle bir Yazılım Ar-Ge Merkezi olan Teracity firmasının ihtiyaçları doğrultusunda geliştirilmiş, sonrasında hem ülkemizdeki diğer Ar-Ge Merkezlerinin hem de tüm şirketlerin kullanabileceği araçlarla zenginleştirilmiş ve satışa sunulmuştur. Ürün temel olarak kişi, faaliyet ve iletişimi yönetmeye yönelik 25 ayrı iş aracı içermektedir.

Aşağıdaki bölümlerde SPICE projesi kapsamındaki 15 sürecin BİLİMP ile nasıl entegre edildiği kısaca sunulacaktır.



Şekil 1. SPICE süreçleri ve seviyeler için gereken süreçler - ISO/IEC TR 15504-Bölüm 7

3.1 Kurumdaki Projelerin Yönetimi

Geliştirilen kurumsal yönetim yazılımı, bakanlığın Ar-Ge Merkezlerinden istediği ve takip ettiği en önemli hususları, çalışmaları ve performans endeksi gibi bilgileri bir sayfada sunan bir Ar-Ge Paneli içermektedir.

Panel tasarımında, Bakanlığın temel aldığı Ar-Ge Merkezleri Performans Endeksi (2015) kullanılmaktadır [6]. Endeks hesaplama bileşenleri, 3 temel bileşen (% 36 Girdi, %26 Süreç ve %38 Çıktı), 6 alt bileşen ile 33 adet göstergeden oluşmaktadır. Bu Ar-Ge Paneli firmadaki büyük resmi vermesi açısından çok önemli bilgiler sunmakta-

dır. Her bir projenin bütün bilgilerinin zamanında ve tam girilmesiyle, her zaman bu panel doğru ve güncel bilgiyi tablolar ve grafiklerle yöneticilere göstermektedir.

3.2 Proje Yönetimi ve Risk Yönetimi

SPICE belgelendirmesi kapsamında şirketimizin "Teraweb-Yerel Yönetimler için Fonksiyonları Arttırılmış Yeni bir E-Belediye v2" ile " Teradesk-Elektronik Belge ve Süreç Yönetim Sistemi v2" projelerimiz incelemeye alınmıştır. Teracity firmasında her bir proje hem bakanlığın istediği proje bilgilerini içerecek hem de proje yönetimini kolaylaştıracak şekilde tanımlanmakta ve takip edilmektedir. SPICE değerlendirmesi öncesinde de uygulanan bu yöntem değerlendirme çalışmaları ile SPICE süreçlerini de içerecek hale getirilmiştir. Bununla ilgili proje tanım ekranına MAN.5 Risk Yönetimi için "Risk Bilgileri" ve SUP.2 Doğrulama süreci için, ISO/IEC TR 15504-Bölüm 5'e göre, yapılan süreç çalışmalarının ayrıntılı bir şekilde doğrulamasını sağlayacak "Süreç Doğrulamaları" alt menüler eklenmiştir.

Proje tanım ekranında "Risk Bilgileri" alt menüsü seçildiğinde, projenin bütün yönetim ve teknik riskleri sıralanmaktadır. "Süreç Doğrulamaları" alt menüsü ise her bir sürecin çıktılarına göre hazırlanan Süreç Doğrulama kriterlerini ve doğrulama işlem sonuçlarını göstermektedir.

3.3 Doğrulama ve Kalite Güvencesi

Teracity'nin Yazılım geliştirmede kullandığı Kalite Yönetim Sistemi, SPICE Seviye-2 SUP.1 Kalite Güvencesi sürecine ve ISO 27001:2013 standardına uygun olarak kurulmuştur.

Bir Süreç Doğrulama (SUP.2) faaliyetinde bulunan uygunsuzluklar, Doğrulama Kaydı ekranında, delili ile birlikte kaydedilir ve bu uygunsuzluk için bir DF (Düzeltici) kaydı açılır. DF kaydının ilgili işlemlerinden bir görev oluşturularak, alınacak önlem kaydına ait faaliyet bilgisi görev açıklaması olacak şekilde Proje Yöneticisine görev olarak atanmaktadır.

Yukarıda kısaca görüldüğü gibi, Proje Yönetimi, Risk Yönetimi, Doğrulama ve Kalite Güvencesinin yaklaşık bütün zaman alan önemli faaliyetlerin tanımlanmaları, görev atamaları, işlerin ve risklerin takibi, kayıtları ve raporlarının alınması, tamamen geliştirilen bu yönetim yazılımı üzerinden yapılmaktadır. Bu ve diğer süreçlerle ilgili, böyle bir yazılım kullanmayan firmalara göre çok az zaman harcanmakta ve az doküman üretilmektedir.

3.4 Dokümantasyon ve Konfigürasyon Yönetimi

Yukarıdaki örneklerde görüldüğü gibi, geliştirilen yönetim yazılımı altyapısının kullanılması, üretilen dokümanların sayısını ve yapılacak işleri çok çok azaltmış ve işlerin takibini kolaylaştırmıştır. Bütün 15 SPICE Seviye 2 süreci için sadece 3 tane ana doküman üretilmiştir. Ana dokümanlardan biri Proje Yönetim Planıdır. Diğer ana doküman, Yazılım Mühendisliği süreçlerinden iki sürecin (ENG.1 Gerekler Toplama ve ENG.4 Yazılım Gerekleri Analizi) gerektirdiği çalışmaları toplayan, Gerekler Toplama ve Analiz Dokümanıdır. Son ana doküman, yine 3 süreci (ENG.5 Yazılım Tasarımı, ENG.6 Yazılım Gerçekleştirme (Kodlama) ve ENG.8 Yazılım Testi) birleştiren, Yazılım Tasarımı ve Gerçekleştirme Dokümanıdır. Ayrıca bu 3 uzun ana dokümana, gereken en az sayıda ekler ilave edilmiştir.

Bu geliştirilen yönetim yazılımı altyapısının kullanılması ve üretilen az sayıda dokümandan dolayı, SPICE Dokümantasyon sürecinin (SUP.7) çıktılarını gerçekleştirmek de kolay ve hızlı olmuştur. Üretilen bütün dokümanlar, analiz, tasarım ve test yönetim yazılımı altyapısında saklanmaktadır. Dokümantasyon sürecinin ana sorumluları Proje Yöneticisi ve Kalite Yöneticisidir.

SUP.8 Konfigürasyon Yönetiminde, yaklaşık bütün proje iş ürünleri için yine ana yönetim yazılımımız kullanılmaktadır. Gereklerin Toplanması ve Analizinde ve Yazılım Tasarımında kullanılan Enterprise Architect ile üretilen dokümanların ve tasarım dosyalarının sürümleri yönetim yazılımı veri tabanında saklanmaktadır. Yazılım kodlarının sürüm takibi için Microsoft Team Foundation Server, kalite aracı olarak Reshaper, fonksiyonel testler için ise Test Cafe yazılımları kullanılmaktadır.

3.5. Problem Çözümü ve Değişiklik Talepleri Yönetimi

Teracity, müşterileri ve Teracity Müşteri Hizmetleri arasında iletişim BİLİMP üzerinden sağlanmaktadır. Bu yapı SPICE Seviye 2'deki Problem Çözümü Yönetimi (SUP. 9) ve Değişiklik Talepleri Yönetimi (SUP.10) süreçlerini karşılamaktadır.

Müşterilerden gelen Kurulum, Hata/Sorun veya Değişiklik talepleri bu altyapı üzerinden alınmakta ve iş akışı yönetilmektedir.

Kuruma gelen tüm müşteri talepleri Şekil 2'de görülen İş Yönetimi Paneli ile takip edilmektedir. Bu panelde müşteriden gelen talepler, tamamlananlar, onaylananlar, bekleyenler ve personel talep dağılımları detaylı şekilde görülmektedir.



Şekil 2. İş yönetimi paneli

4. Süreç Performans Hedefleri ve Ölçümleri

Geliştirilen yönetim yazılımının üst yönetime sağladığı en önemli desteklerinden biri, performans tabanlı süreç yönetimidir. Kurumdaki bütün işlerin ve projelerin yönetiminde kullanılan bu yazılımda, bütün yönetim verileri veri tabanında sürekli saklandığı için, en yukarıdan Ar-Ge Merkezi Performans hedefleri ve düşük seviyeli SPICE süreç performans hedeflerinin bu verilere göre tanımlanması, performans hedeflerinin belirlenmesi ve hedeflerden sapmalarda gerekli önleyici faaliyetlerin düzenlenmesi sağlanmaktadır. Yukarıda Bölüm 2 de anlatılan Ar-Ge Paneliyle takip edilen ve yönetilen üst düzeydeki performans hedefleri, genelde kurumun en üst yönetimini ilgilendirmesinin yanında, SPICE ile tanımlanan Proje Yönetim ve Destek ve Mühendislik performans hedefleri, proje yöneticilerini ve proje elemanlarını daha çok ilgilendirmektedir. Aşağıdaki Tablo 1'de, SPICE Seviye 2'deki 15 süreç için, tanımlanabilecek Performans Hedefleri için bazı örnekler verilmiştir. Hedeflerin çoğu veri tabanında tutulan veriler göz önüne alınarak, sayısal olarak tanımlanmakta, takip edilmekte izlenmektedir. Hedeflenen performans verileri, projenin geliştirilmesi boyunca ölçülmekte ve tutturulamaması durumunda, gereken uyarılar zamanında verilerek, riskler önlenmektedir.

Tablo 1. SPICE seviye 2 süreç performans hedefleri için örnekler

Proje Yönetimi	Düzeltici/önleyici faaliyet kapsamında tanımlanan faaliyetlerin %90'ının belirtilen termin tarihinden önce tamamlanması
	Projenin hedeflenen süreden en fazla %10 sapma ile tamamlanması
Risk Yönetimi	Risk değeri yüksek ve tolere edilemez olan risklerin sayısının tüm risklerin sayısına oranının %10'u aşmaması
	Risk seviyesini aşağı çekme amaçlı olarak açılan düzeltici/önleyici faaliyetlerin %90'ının belirtilen termin tarihinden önce tamamlanması
Kalite Güvence	Müşteri memnuniyetinin en az %80 olması
	Müşteriden gelen taleplerin oran olarak en fazla %20'sinin "hata" kaynaklı talep olması
Doğrulama	Doğrulama denetim sonuçlarındaki uygunsuzluk oranının %15'i aşmaması
	Düzeltici/önleyici faaliyet kapsamında tanımlanan faaliyetlerin %90'ının belirtilen termin tarihinden önce tamamlanması
Dokümantasyon	Yılda bir kez doküman gözden geçirme yapılması
	Düzeltici/önleyici faaliyet kapsamında tanımlanan faaliyetlerin %90'ının belirtilen termin tarihinden önce tamamlanması
Konfigürasyon Yönetimi	Her süreç doğrulama işlemindeki doğrulama adımlarının en fazla %15 olumsuzluk ile sonuçlanması

	Olumsuz sonuçlanan doğrulama adımları için açılan düzeltici/önleyici faaliyetlerin %90'ının termin tarihinden önce sonuçlandırılması
Problem Çözüm ve Talep ve Değişiklik Yönetimi	Taleplerin %90'ının üzerlerindeki termin tarihinden önce tamamlanması
	Kurum onayına alındıktan sonra kurumdan geri dönen taleplerin oranının %5'in altında olması
	Ortalama talep kapatma süresinin 21 iş günü aşmaması
Gereksinimlerin Toplanması ve Analizi	Talep niteliği yeni çalışma/geliştirme olan taleplerde talebin analiz biriminde kalma süresinin talebin açılış tarihinden kapatılış tarihine kadar geçen sürenin %30'unu geçmemesi
Yazılım Tasarımı	Tüm tablolarda primary key bulunması
	Yazılım gözden geçirmelerde görülen uygunsuzluklar kapsamında açılan DÖF kayıtlarındaki faaliyetlerden %90' ı termin tarihinden önce tamamlanması
	Projelerdeki kütüphaneler karşılıklı olarak bağımlı olmaması
Yazılım Yapılandırma	Yazılım gözden geçirme toplantılarında görülen uygunsuzluklar kapsamında açılan DÖF kayıtlarındaki faaliyetlerden %90'ı termin tarihinden önce tamamlanması
Yazılım Entegrasyonu	Müşteriden gelen taleplerin oran olarak en fazla %20'si hata talebi olması
Yazılım Testi	Müşteriden gelen taleplerin oran olarak en fazla %20'si hata talebi olması
	Kurum onayına alındıktan sonra kurumdan geri dönen taleplerin oranının %5 in altında olması
Sürüm Yönetimi	Yeni sürüm için güncelleme paketi hazırlama işi sorumlu kişi için zaman maliyeti olarak yarım saati aşmamalı
	Sürüm güncellemesi sonrası herhangi bir sebepten ötürü bir önceki sürüme dönme işi sorumlu kişi için zaman maliyeti olarak yarım saati aşmamalı

5. Sonuçlar ve Tartışmalar

Şubat 2017 ile Ekim 2017 tarihleri arasında yaklaşık 8 ay süren, Teracity'de yapılan süreçleri iyileştirme çalışmalarında, SPICE süreçlerinin önemli bir kısmının firmanın iş yönetim yazılımı ile entegrasyonu sağlanmıştır. Bu sayede, daha önceden firmada yaşanan iş ve proje geliştirme süreçleri SPICE desteğiyle daha da iyileştirilmiştir. Ayrıca bu değişim kolay ve hızlıca yapılmış, çalışanlar tarafından kısa sürede benimsenmiştir. SPICE'ın bu iş yazılımına eklenmesiyle geliştirdiğimiz yeni iş modeli ve SPICE için gereken bütün süreç çıktılarının üretilmesi ve işlerin değerlendirme için tamamlanması 8 ay sonunda, bu konuda herhangi bir danışmanlık alınmadan tamamen şirket personelinin yaptığı çalışmalar ile SPICE belgesinin de alınmasıyla başarılı bir şekilde bitirilmiştir. SPICE için sağlanan çıktılar, geliştirilen iş modeli ve yönetim yazılımı altyapısı sayesinde normalde ülkemizde 4 gün süren SPICE değerlendirmesi, Teracity'de 2,5 günde tamamlanmıştır. Kurumdaki çalışma süreçlerinin çalışanlar tarafından benimsenerek, kolay, hızlı, tabii değişimi ve olgunluk seviyesinin belgelendirmesinin de hızlı ve kolay olması, başarılı bir süreç iyileştirme çalışmasının en önemli hususlarındandır.

SPICE Seviye 2'nin her bir sürecinin kurum içinde ne şekilde ve hangi araçlarla sağlandığının özeti aşağıda Tablo 2'de yer almaktadır.

Tablo 2. SPICE süreçleri ve kullanılan araçlar

SPICE Süreçleri	Kullanılan Araç	Açıklama
MAN.3 Proje Yönetimi	BİLİMP	Ar-Ge Proje Yönetim aracı ile sağlanmaktadır. Her bir proje tanımlandığı esnada riskleri ve alınacak önlemleri de belirlenmekte ve izlenmektedir. Doğrulama süreci de her projenin kendi içinde yapılmaktadır.
MAN 5. Risk Yönetimi	BİLİMP	
SUP.2 Doğrulama	BİLİMP	
SUP.1 Kalite Güvencesi	BİLİMP	Kalite aracı ile sağlanmaktadır. Doğrulama sonrası tespit edilen uygunsuzluklar düzeltici faaliyetlerle takip edilmektedir.
SUP.7 Dokümantasyon	BİLİMP	Teknik çizim içeren dokümanlar harici tüm dokümanlar bilimp'e girilen verilerle sağlanmaktadır.
SUP.8 Konfigürasyon Yönetimi	BİLİMP	İş yönetimi aracı ile sağlanmaktadır. Talep numaraları, talebe ilşkin yapılan geliştirmeler ve oluşturulan versiyon paketleri bu araç ile izlenmektedir. Konfigürasyon yönetiminin iyileştirilmesi için TFS ile entegrasyon planlanmaktadır.

SUP.9 Problem Çözümü Yönetimi	BİLİMP	İş yönetimi aracı ile sağlanmaktadır.
SUP.10 Değişiklik Talep Yönetimi	BİLİMP	İş yönetimi aracı ile sağlanmaktadır.
ENG.1 Gerekler Toplama	BİLİMP	Bilimp iş yönetimi aracı ile sağlanmaktadır.
ENG.4 Yazılım Gerekleri Analizi	BİLİMP+Diğer	Bilimp iş yönetimi aracı ve enterprise architect ile sağlanmaktadır.
ENG.5 Yazılım Tasarımı	BİLİMP+Diğer	
ENG.6 Yazlım Gerçekleştirme (Kodlama)	Diğer	Microsoft Team Foundation Server ve kodlama araçları kullanılmaktadır.
ENG.7 Yazılım Entegrasyonu	Diğer	Microsoft Team Foundation Server ve kodlama araçları kullanılmaktadır.
ENG.8 Yazılım Testi	Diğer	Test cafe kullanılmaktadır.
SPL.2 Ürün Teslim	BİLİMP	Bilimp iş yönetimi aracı ile sağlanmaktadır. Testler sonrası oluşturulan sürüm paketleri haritalama aracı ile müşterilere güncellenmektedir.
Süreç Performanslarının Ölçümü ve Değerlendirme	BİLİMP	Her bir süreç için tanımlı performans hedeflerine ait ölçümler Bilimp esnek raporlama aracı ile yapılmaktadır. Sapmalar düzeltici faaliyetlerle takip edilmektedir.

SPICE destekli BİLİMP yazılımıyla (BİLİMP-SPICE) sağlanan temel Yönetim Bilgi Sistemi fonksiyonlarının, yurtdışı örneklerinin sağladıklarıyla karşılaştırdığımızda, farklı yazılımlarda olan en önemli temel fonksiyonları içerdiğini görmekteyiz. Örneğin, yönetim yazılımları alanının liderlerinden olan Atlassian [7] ailesinin ürünleriyle BİLİMP'i karşılaştırdığımızda, en az 4 farklı ürünündeki (JIRA Core, JIRA Service Desk, JIRA Software ve JIRA Status) en temel fonksiyonları BİLİMP'in içerdiğini görmekteyiz. Tablo 3'de sadece 3 Atlassian JIRA ürününün BİLİMP-SPICE ile karşılaştırması verilmiştir.

Tablo 3. BİLİMP-SPICE'ın JIRA ve SPICE Değerlendirme araçlarıyla karşılaştırması

	JIRA Software	**JIRA SERVICE**	**JIRA CORE**	**SPICE 1-2-1**	**SPICE LITE**	**BİLİMP-SPICE**

		DESK				
Kurum Yönetimi	-	-	Kısmi	-	-	Tam
Projelerin Yönetimi	Kısmi	-	Tam	-	-	Tam
Proje Yönetimi	Tam	-	-	-	-	Tam
Problem Çözümü ve Değişklik Talebi Yönetimi	Kısmi	Tam	-	-	-	Tam
SPICE Desteği	-	-	-	Tam	Tam	Kısmi
SPICE Süreç Değerlendirmesi	-	-	-	Tam	Tam	Kısmi
Ar-GE Merkezi Yönetimi	-	-	Kısmi	-	-	Tam

JIRA Core'da bulunan, Projelerin Yönetimi, İnsan Kaynakları (HR), Yasal (Legal) ve Mali Durum (Finance) gibi fonksiyonlar BİLİMP'te mevcuttur. Örneğin, 5746 ve 5510 sayılı kanunları içeren maaş hesaplamaları gibi mali işlemler ve insan kaynaklarının tümünü kapsayan işlemler BİLİMP'te bulunmaktadır. JIRA Service Desk'teki, müşteri isteklerinin alınması ve bu taleplerin bir analiz ilk ön işleminden geçtikten sonra, ilgili projelere iletilmesi, BİLİMP'te çok güçlü bir şekilde sağlanmaktadır ve bu aynı zamanda SPICE Seviye 2'deki iki süreci ((SUP.9 ve SUP.10) karşılamaktadır. JIRA Software'ın fonksiyonlarının önemli bir kısmı, yine BİLİMP'te mevcuttur. JIRA Status Page'teki Performans Metrikleri, Son-Kullanıcı Bildirimleri, Sayfaların Kişiselleştirilmesi, Özel Sayfalar ve Entegrasyon gibi fonksiyonlar da BİLİMP'te bulunmaktadır. BİLİMP-SPICE'ın sağlamış olduğu Ar-Merkezlerine yönelik yönetim fonksiyonları, performans endeks hesaplaması ve SPICE desteği JIRA ürünlerinde bulunmamaktadır.

SPICE veya CMMI modelleriyle süreç iyileştirme çalışmalarında, genellikle çok fazla sayıda doküman üretilmekte ve süreç yönetiminde bir yazılım kullanılmamaktadır. Süreç iyileştirme ve değerlendirmeye yönelik ticari veya akademik bazı ücretsiz yazılımlar az da olsa olmasına rağmen, bu yazılımlar sadece CMMI veya SPICE süreçleriyle ilgili olup bu yazılımlarla bir kurumun iş süreçleriyle bağlantısının kurulup, iş ve proje yönetimin yapılması mümkün değildir. Halbuki, geliştirilen BİLİMP-SPICE yönetim yazılımı, kurumsal yönetimin yanında, iş ve proje yönetimini SPICE süreçlerine ve Ar-Ge Merkezleri için bakanlık gereksinimlerine göre yaparak yönetimsel anlamda kolaylık sağlamaktadır.

İlerleyen zamanda Bakanlık tarafından yayınlanan performans endekslerinde olabilecek değişikliklerin veya ar-ge mevzuatındaki değişikliklerin yazılım sürüm güncellemeleri ile yönetilmesi planlanmaktadır.

Bölüm 2 'de anlatılan SPICE ölçüm çerçevelerinden Seviye II "Planlı" adımından Seviye III "İyi tanımlanmış şekilde işler" seviyesine geçiş ile ilgili yapılacak çalışma-

lar için BİLİMP üzerinde geliştirmeler yapılacak ve ilave süreçlerin de buradan yönetilmesi sağlanacaktır.

Kaynakça

1. Software Engineering Institute (SEI) (2010), CMMI for Development, v1.3.
2. ISO/IEC 15504-5:2006 Information technology — Process Assessment — Part 5: An exemplar Process Assessment Model.
3. ISO/IEC TR 15504-7:2008 Information technology — Process assessment — Part 7: Assessment of organizational maturity.
4. Singh R., International Standard ISO/IEC 12207 Software Life Cycle Processes, Software Process Improvement and Practice, 2(1), 35-50, 1996.
5. BİLİMP Yönetim Bilgi Sistemi yazılımı: http://www.bilimp.com/.
6. Ar-Ge Merkezleri Performans İndeksi (2015), Bilim Teknoloji ve Sanayi Bakanlığı.
7. Atlassian: https://www.atlassian.com/

Alt Yüklenici Yönetiminde Çevik Bir Yaklaşım Deneyimi

Gonca Hülya SELÇUK, Hakime KOÇ

ASELSAN A.Ş., Ankara

ghselcuk@aselsan.com.tr
unsal@aselsan.com.tr

Özet. Günümüzde yazılım geliştirme projelerinde yaygın olarak kullanılan çevik yöntemler, ihtiyaçları sık değişen projelerdeki başarı oranını artırmaktadır. Yazılım geliştirme projelerinde kullanılan çevik yaklaşımlar, ihtiyaçlar doğrultusunda tek başına veya birden fazla yaklaşımın harmanlanması ile bir arada kullanılabilmektedir. Alt yükleniciler tarafından geliştirilen yazılımlar ya da yazılım bileşenleri kapsamındaki alt yüklenici yönetimi faaliyetlerinin de yazılım geliştirme faaliyetlerini destekleyerek proje başarısına katkı sağlayacağı öngörülmektedir. Çevik yazılım geliştirme faaliyetlerinin çevik alt yüklenici yönetimi ile desteklenebilmesi adına; bu çalışmada alt yüklenici yönetimi için çevik bir yaklaşım önerilmektedir. Tecrübelerin de paylaşılacağı bu çalışmada, önerilen çevik yaklaşımın uygulanmasının projeye katkıları ve geleceğe dair çalışma önerileri sunulmaktadır.

Anahtar Kelimeler: Alt Yüklenici Yönetimi, Çevik Yaklaşımlar, Çevik Yazılım Geliştirme.

Experience with Agile Approach in the Subcontractor Management

Gonca Hülya SELÇUK, Hakime KOÇ

ASELSAN A.Ş., Ankara

ghselcuk@aselsan.com.tr
unsal@aselsan.com.tr

Abstract. Nowadays, agile methods, which are widely used in the software development projects, increase the success rate of projects, where requirements are changing very frequently. Agile approaches used in the software development projects can be used alone or in combination with multiple approaches according to needs. It is also expected that subcontractor management activities will contribute to the success of projects in which all software or some components of

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

software developed by subcontractors. In this study, an agile approach for subcontractor management is proposed to support all agile development activities. Moreover, this study presents the application of this agile approach in a specific industrial context with its benefits to all stakeholders in subcontractor activities.

Keywords: Subcontractor Management, Agile Approaches, Agile Software Development.

1 Giriş

Yazılım geliştirme projelerinde çevik yaklaşımlar günümüzde yaygın olarak kullanılmaktadır. Çevik yaklaşımlar, detaylandırılamamış gereksinimler içeren projelerde zaman içerisinde iterasyonlar halinde çalışılabilmesine imkan tanımakta, dağıtık çalışan takımların çalışmalarını desteklemektedir. Belirsizliklerin yönetimine ve etkileşimin artmasına katkı sağlayan çevik pratikler projelerin başarısını da etkilemektedir. Standish Group International'ın 2015 yılında 10.000'i aşan proje ile yapmış olduğu çalışmada, çevik süreçler kullanılan projeler ile şelale modeli kullanılan projelerdeki başarı oranı karşılaştırılmıştır [1]. Çalışma sonuçlarına göre, çevik süreç kullanılan projelerin başarı oranı şelale modeli kullanılan projelerin başarı oranının dört katıdır.

Şirketler yazılım geliştirme faaliyetlerini kendi bünyelerinde gerçekleştirebildikleri gibi tedarik etmeyi de farklı nedenlerle yoğun olarak tercih etmektedir. Savunma sektörü projelerinde de alt yüklenici kullanımı önemli bir paya sahiptir. Türk Silahlı Kuvvetlerini Güçlendirme Vakfı Bağlı Ortaklıkları tarafından 2014 yılsonu itibarıyla 3.230 alt yüklenici ile çalışılmaktadır [2]. Yazılım geliştirme projelerinin sözleşmeler kapsamında tedarik edildiği ve alt yükleniciler tarafından gerçekleştirildiği durumlarda, proje yönetim faaliyetleri yanı sıra alt yüklenici yönetim faaliyetleri de projenin başarısını etkileyen önemli unsurlardan biri olmaktadır. Farklı ihtiyaçlara sahip projelerde farklı yönetim yaklaşımları öne çıkmaktadır [3]. Bu doğrultuda, çevik süreçlerle geliştirilen ve yönetilen projelerde alt yüklenici yönetiminin de çevik yaklaşımlara uyum sağlaması önem taşımaktadır [4].

ASELSAN Radar ve Elektronik Harp Sistemleri Sektör Başkanlığı (REHİS) bünyesinde alt yükleniciler tarafından çevik süreçler kullanılarak geliştirilmekte olan projeler yürütülmektedir. İlgili projeler kapsamında ASELSAN REHİS tarafından gerçekleştirilen alt yüklenici yönetimi faaliyetleri için çevik bir yaklaşım tanımlanmıştır. Bu bildiride, alt yüklenici yönetimi için tanımlanan çevik yaklaşıma yer verilmektedir. Bölüm 2'de çevik yaklaşımlar üzerine genel bilgi verilmektedir. Bölüm 3 önerilen çevik yaklaşımı tanımlamakta; Bölüm 4 ise tanımlanan modelin bir ürün hattı yazılım geliştirme projesindeki uygulamasını özetlemektedir. Bölüm 5; sonuçları, öğrenilen dersleri ve gelecek çalışmalar için önerileri içermektedir.

2 Çevik Yaklaşımlar

Günümüzde yazılım geliştirme faaliyetlerinde yoğunlukla tercih edilen çevik yaklaşımlar farklı şekillerde kullanılabilmektedir. Bu yaklaşımların odağındaki "çevik"lik;

- Bireyler ve etkileşim
- Çalışan yazılım üretme
- Müşteri ile işbirliği içinde çalışma
- Değişime cevap verme

değerlerini öne çıkarmaktadır [5].

Çevik kavramını hayata geçiren yaklaşımlardan biri olan Scrum, ilk defa 1995 yılında tanıtılmıştır. Scrum, karmaşık ürünlerin geliştirilmesini, teslim edilmesini ve sürdürülebilir olmasını bir çerçeve (framework) olarak tanımlamaktadır [6]. Bu çerçeve yinelemeli ve artırımlı bir şekilde, kullanılabilir ürün geliştirmeyi desteklemektedir. Özünde Scrum takımı olarak adladırılan, küçük takımlar yer almaktadır. Scrum takımı; bir ürün sahibi (*product owner*), geliştirme takımı ve Scrum Master'dan oluşmaktadır. Scrum takımının rol ve sorumlulukları aşağıdaki tabloda yer almaktadır.

Tablo 1. Scrum Takımı Rolleri

Rol	Sorumluluk
Ürün Sahibi	Ürünün veya projenin iş listesini (*product backlog*) yönetmekten sorumludur.
Geliştirme Takımı	Her bir iterasyonda (*sprint*) "Bitti (*Done*)" tanımına uyan ve yayınlanabilir bir iş parçasını teslim etmekten sorumludur.
Scrum Master	Scrum'ın tanımlanmış çerçeveye uyumlu yürütüldüğünü garanti etmekten sorumludur.

Scrum kapsamında "çevik"liğin desteklenmesine yönelik olarak İterasyon (Sprint), İterasyon Planlama, Günlük Değerlendirme, İterasyon Değerlendirme, İterasyon Retrospektifi etkinlikleri de tanımlanmıştır. Roller ve etkinliklerin yanı sıra aşağıdaki unsurlar da tanımlanmıştır:

- İş Listesi (*Product Backlog*): Üründe ihtiyaç duyulan her şeyi içeren bir listedir.
- İterasyon İş Listesi (*Sprint Backlog*): İterasyon kapsamında yapılmasına karar verilen ve "Ürün İş Listesi"nden seçilen bir listedir.
- Ürün Parçası: İterasyon sonucunda, mevcutta gerçekleştirilen tüm iterasyonlardaki işlerin "Bitti (*Done*)" kavramına eriştiği, ürünün belirli özelliklerle çalışan bir parçasıdır.
- "Bitti (*Done*)" Kavramı: Scrum takımında şeffaflığın sağlanarak işin hangi durumda bitti sayılabileceğinin tanımını içermektedir.

Scrum çerçevesinin yanında Kanban yaklaşımı da yazılım geliştirmede 2000li yılların başından itibaren öne çıkan yaklaşımlardan biridir. Kanban, ilk defa 1940lı yıllarda Japonya'da yalın üretim (*lean production*) ilkesiyle ortaya çıkmıştır. Kelime anlamı ile "görsel sinyal" veya "kart" demek olan Kanban, üretimde her aşamada kullanılan kartlara karşılık gelmektedir. Yalın üretim ilkesinden yola çıkılarak modellenen Kanban 2000li yıllarda yazılım geliştirme alanında da kullanılmaya başlanmıştır [7]. Kanban'ın çıkış noktasında süreç iyileştirme yer alırken yazılım geliştirme alanında;

- İş akışının görselleştirilmesi ve yönetilmesi
- Aynı anda gerçekleştirilen iş sayısının limitlenmesi
- Süreçlerin açık hale getirilmesi

olarak özetlenebilecek üç temel prensibi benimsemektedir. İş akışının görselleştirilmesi ve tüm takımın erişebilmesi sonucunda, bir takım üyesi bir işi bitirdiğinde sıradaki bir sonraki işe geçmesi hedeflenmektedir.

Scrum ile Kanban yaklaşımları, temel olarak işlerin çevik bir şekilde tamamlanmasını hedeflemektedir. Farklılık taşıdıkları noktalar Tablo 2'de özetlenmektedir. Scrum çerçevesindeki tanımlı kurallar (roller, etkinlikler) yerine Kanban kullanımının verimliliği artırdığını savunan örnekler [8] bulunduğu gibi; farklılıkların yanı sıra Scrum ve Kanban yaklaşımlarının beraber kullanıldığı "Scrumban" gibi yaklaşımlar da bulunmaktadır [9].

Tablo 2. Scrum ve Kanban Farklılıkları

Scrum	Kanban
Geliştirme zaman limiti dahilinde, iterasyonlarla yapılır.	Geliştirme süreklidir, zaman limiti konulması zorunlu değildir.
Süreç iyileştirme aktiviteleri tanımlıdır.	Süreç iyileştirme aktiviteleri için belirli bir tanım yoktur.
Roller tanımlıdır.	Herhangi bir rol tanımlı değildir.
İterasyon iş listesi her iterasyon başında oluşturulur.	İş listesi sürekli güncellenerek kullanılır.
İşlerin tamamlanmasındaki temel hedef, kısa aralıklar dahilinde gerçekleştirilmesidir.	İşlerin tamamlanmasındaki temel hedef, bir akış içerisinde gerçekleştirilmesidir.

3 Alt Yüklenici Yönetiminde Çevik Yaklaşım İhtiyaçları

Alt yüklenici yönetimi kapsamında, savunma sektöründe bir ürün hattı yazılım geliştirme projesindeki çeşitli ihtiyaçlardan yola çıkılarak çevik bir yaklaşım önerisi oluşturulmuştur. Çevik yaklaşımın oluşturulma motivasyonu aşağıdaki konu başlıklarında özetlenmektedir. Bu kapsamda, yazılım geliştirme projelerini gerçekleştiren proje ekibi "Alt Yüklenici", projeye ihtiyaç duyarak alt yüklenici aracılığıyla tedariğini sağlayan ve yöneten ekip de "Alt Yüklenici Sorumlusu" olarak adlandırılmaktadır.

3.1 Gereksinim Mühendisliği / Ürün Sahipliği

Alt Yüklenici tarafından geliştirilmekte olan projede çevik bir yaklaşım olan Scrum [6] kullanılmaktadır. Scrum pratikleri kapsamında, yazılım geliştirme faaliyetlerinin içeriğini belirlemek için "iş listesi (*product backlog*)" Alt Yüklenici ve Alt Yüklenici Sorumlusu tarafından tanımlanmaktadır. İş listesinde, detaylı yazılım gereksinimlerini oluşturabilecek seviyede işler bulunabildiği gibi detaylandırılma ihtiyacı bulunan gereksinimler de yer almaktadır. İhtiyaç sahibi firmada gereksinim mühendisliği faaliyetlerinin gerçekleştirilerek kullanım senaryolarının detaylandırılması veya oluşturulmasına ihtiyaç duyulmaktadır. Kullanım senaryoları, iş listesine girdi sağlamakta ve iş listesinin detaylandırılmasını sağlamaktadır. Alt Yüklenici tarafından kullanılan Scrum pratikleri kapsamındaki "ürün sahibi (*product owner*)" rolü gereksinim mühendisliği faaliyetleri açısından yeterli olamamaktadır. Bu noktada ürün sahibinin;

- ihtiyaç sahibi firmadaki matris yapılanma nedeniyle birden fazla bölümden bilginin toplanması,
- proje kapsamındaki ihtiyaçların sık değişmesi,
- projenin entegre edileceği birim veya proje sayısının fazla olması,
- diğer projelerin gereksinimlerinin proje ihtiyaçlarında etkili olması

gibi faktörleri yönetmesi gerekmektedir. Proje için belirtilen bu faktörler ve yapılmış çalışmalar doğrultusunda, Alt Yüklenici Sorumlusu tarafından da ihtiyaç sahibi firma içerisindeki "ürün sahibi (*product owner*)" rolünün üstlenilmesi gerekliliği ortaya çıkmaktadır.

3.2 İşlerin Planlanması ve Takibi

Alt Yüklenici tarafından yazılım geliştirme faaliyetleri iterasyonlar bazında gerçekleştirilmektedir. Her bir iterasyon öncesinde Alt Yüklenici tarafından gereksinim mühendisliği / ürün sahipliği faaliyetleri sonucu elde edilen detaylı senaryolar, gereksinimler veya faaliyetler değerlendirilmekte; yeterli detayı içeren ve öncelikli olan işler seçilerek iterasyon kapsamına alınmaktadır. Alt Yüklenici tanımladığı iterasyonlar kapsamındaki faaliyetlerini üç haftalık zaman dilimleri içerisinde tamamlamayı hedeflemektedir. Bu kapsamda; Alt Yüklenici Sorumlusu faaliyetlerinin niteliği ise Alt Yüklenici faaliyetlerine göre farklılık göstermekte, işler Alt Yüklenici'ye göre daha genel çerçeveden ve detay içermeden takip edilmektedir. Alt Yüklenici Sorumlusu tarafından takip edilen işler ve yürütülen faaliyetler proje açısından zaman kısıtı taşısa da, belirlenen bir zaman limiti içerisinde başlayıp bitirmek mümkün olamamaktadır. Bu nedenle Alt Yüklenici Sorumlusu iş ve faaliyetlerinin zaman limitinden veya iterasyonlardan bağımsız olarak takip edilmesi ihtiyacı bulunmaktadır.

3.3 Entegrasyon Ortamı

Alt Yüklenici ile Alt Yüklenici Sorumlusu fiziksel olarak farklı lokasyonlarda çalışmaktadır. Geliştirilen projenin gizlilik seviyesi nedeniyle geliştirme Alt Yüklenici lokasyonunda, kapalı ağ ortamında gerçekleştirilmektedir. Bu doğrultuda, proje konfigürasyon ortamına fiziksel olarak Alt Yüklenici Sorumlusu tarafından erişilememektedir. Ancak, alt yüklenici faaliyetlerinin eş zamanlı ve etkileşimli yönetilebilmesi açısından, Alt Yüklenici ve Alt Yüklenici Sorumlusu arasında sürekli bir iletişim ve entegrasyon ortamının kurulabilmesi önem taşımaktadır. Projenin gizlilik kısıtları entegrasyon ortamını da etkilemekte Alt Yüklenici tarafından kullanılan çevik proje yönetimi ortamına (iş listesini, tanımlanan işleri, faaliyetleri, değişiklik isteklerini, hataları içeren ortam) erişilememektedir. Çevik yaklaşımların farklı lokasyonlarda uygulanmaya çalışıldığı durumlarda yüzyüze iletişim kurulamadığından, proje ile ilgili tüm bilgilerin (mevcut durum, planlama, gereksinimler, kullanım senaryosu vb.) dokümantasyonun iyi bir şekilde yapılması ve yazılı iletişimin sağlanması önem kazanmaktadır [10, 11]. Bu nedenle, Alt Yüklenici Sorumlusu ortamında da tüm bilgilerin dokümante edilebilmesi ve Alt Yüklenici yönetim faaliyetlerinin üst seviye takip edilebilmesi için çevik bir proje yönetim ortamının kurulması önemli hale gelmektedir.

3.4 Doğrulama

Doğrulama (*verification*) faaliyetleri Alt Yüklenici bünyesinde gerçekleştirilmekte olan iç bir süreçtir. Alt Yüklenici tarafından uygulanan çevik süreçler kapsamında yazılım geliştirme faaliyetleri için "Bitti (*Done*)" kavramı tanımlanmıştır. "Bitti" kriterlerinden biri de gerçekleştirilmiş olan faaliyetin doğrulanmasıdır. Alt Yüklenici Sorumlusu tarafından iterasyon sonu tanıtımlarına katılım sağlanmakta ancak geçerlileme (*validation*) faaliyetleri birden fazla iterasyon sonucunda oluşturulan sürüm kabul faaliyetleri sonucunda yapılmaktadır. Hataların daha erken bir aşamada yakalanabilmesi için Alt Yüklenici Sorumlusu'nun geçerlileme faaliyetlerinden önceki bir aşamada doğrulama faaliyetlerine katkıda bulunması ihtiyacı bulunmaktadır.

4 Alt Yüklenici Yönetiminde Çevik Bir Yaklaşım

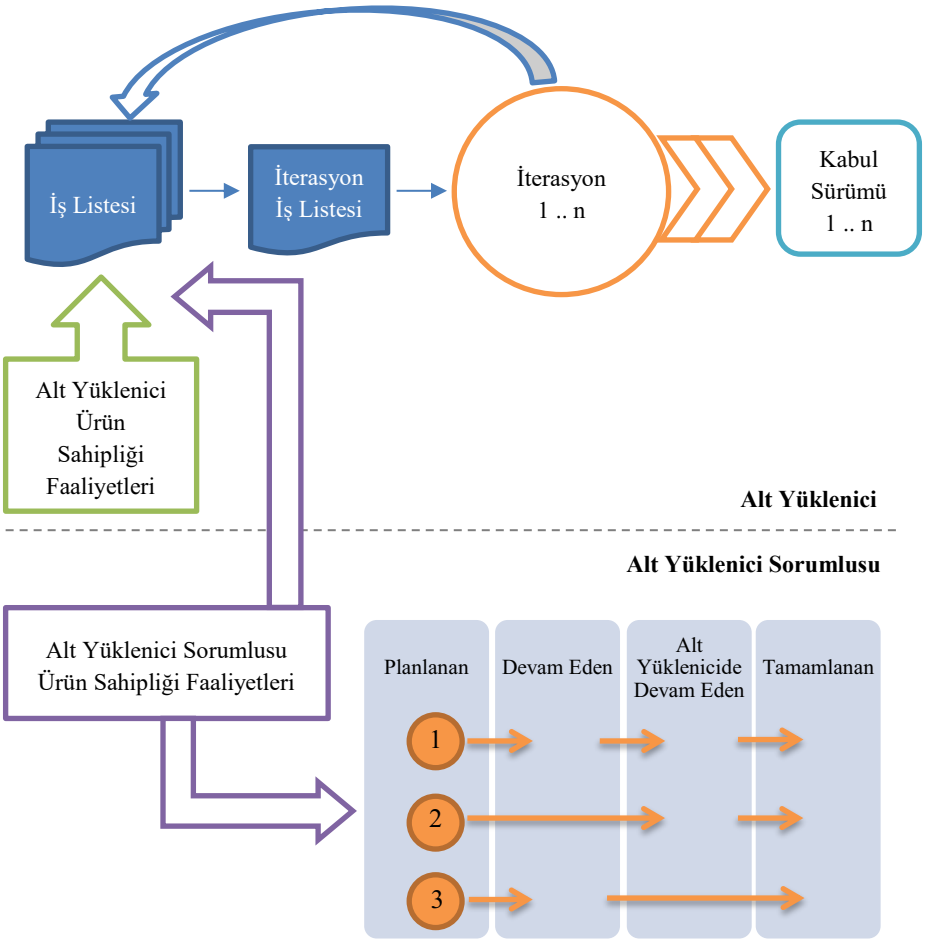
İhtiyaçlardan yola çıkılarak, Kanban ve Scrum yaklaşımları özelliklerinden faydalanılarak alt yüklenici yönetimi faaliyetleri için çevik bir yaklaşım modellenmiştir (Şekil 1). Bu modelde; alt yüklenici yönetimi iş ve faaliyetlerinin gerçekleştirilmesinde iterasyon tanımlaması yapılmadan Kanban tahtası kullanılması önerilmektedir. İterasyon tanımlanmasa da planlamayı kolaylaştırması açısından tahtada son iki ayın iş ve faaliyetlerine odaklanılmaktadır.

Birden fazla alt yüklenicideki Scrum takımını yöneten bir firmada, tüm Scrum takımlarına bilgi aktarımının yapılması için "Alan Ürün Sahibi (*Area Product Owner*)" rolünün oluşturulması önerilmektedir [12]. Birden fazla projenin yürütülebileceği öngörülerek Kanban tahtası birden fazla projeye ait iş maddesi içerebilmektedir. Gereksinim mühendisliği faaliyetleri ihtiyaç sahibi firmada da gerçekleştirilerek, Alt Yüklenici Sorumlusu'nun "ürün sahipliği" rolünü de üstlenmesi önerilmektedir. Bu noktada, Alt Yüklenici Sorumlusu'nda "ürün sahibi"nin, güncel kullanım senaryolarını Alt Yüklenici'deki "ürün sahibi"ne aktarması planlanmaktadır.

Alt yüklenici yönetimi kapsamında yapılması gereken iş maddelerinin bulunabileceği dört farklı durum tanımlanmaktadır: Planlanan, Devam Eden, Alt Yüklenicide Devam Eden, Tamamlanan. "Planlanan" durumdaki işler henüz başlamamış ancak iki ay içerisinde başlaması öngörülen işlerdir. "Devam Eden" durumdaki işler Alt Yüklenici Sorumlusu tarafından planlanmış ve üzerinde çalışılmaya başlamış olan işlerdir. "Alt Yüklenicide Devam Eden" durumdaki işler Alt Yüklenici Sorumlusu tarafından detaylandırılan kullanım senaryoları kapsamında Alt Yüklenici'ye aktarılmış olan işlerdir. "Tamamlanan" durumdaki işler ise Alt Yüklenici veya Alt Yüklenici Sorumlusu tarafından bitirilmiş olan işlerdir. Tamamlanan bir iş Alt Yüklenici tarafından gerçekleştirilmiş ise Alt Yüklenici Sorumlusu tarafından resmi olmayan bir doğrulama işlemi de gerçekleştirilmektedir.

Bir iş maddesinin durumlarına göre ilerleyebileceği üç farklı iş akışı tanımlanmaktadır:

1. Planlanan, Devam Eden, Alt Yüklenicide Devam Eden, Tamamlanan
2. Planlanan, Alt Yüklenicide Devam Eden, Tamamlanan
3. Planlanan, Devam Eden, Tamamlanan



Şekil 1. Çevik Yaklaşım Önerisi Modeli

5 Alt Yüklenici Yönetiminde Çevik Yaklaşım Deneyimleri

Alt yüklenici yönetimi için modellenen çevik yaklaşım savunma sektöründeki bir ürün hattı yazılım geliştirme projesinde uygulamaya alınmıştır. Mevcut durumda, projede yazılım geliştirme faaliyetlerine devam edilmektedir. Ürün hattı projesinin çıktısı iki farklı projede kullanılmaktadır. Ürün hattı yazılım geliştirme projesi ile ürün hattını kullanan iki projenin yazılım geliştirme faaliyetleri eş zamanlı olarak Alt Yüklenici Sorumlusu tarafından takip edilmektedir.

Çevik yaklaşım önerisi, projelerin yazılım geliştirme süreci devam ederken hayata geçirilmiştir. Alt Yüklenici Sorumlusu tarafından gerçekleştirilen iş ve faaliyetler çevik bir proje yönetimi ortamında takip edilmeye başlanmıştır. Kullanılan çevik proje yöne-

tim ortamında halihazırda yürütülen işler ve planlanan işler tanımlanmıştır. İşlerin kapsamı, Alt Yüklenici işlerini kapsayacak şekilde kurgulanarak entegrasyon ortamı Alt Yüklenici Sorumlusu'nda da oluşturulmuştur. Öneri doğrultusunda çevik proje yönetimi aracı üzerinde iki aylık işleri gösteren bir filtre tanımı yapılmıştır. Alt Yüklenici Sorumlusu "ürün sahibi" rolünü de üstlenerek, ihtiyaç sahibi firma içerisindeki gereksinim mühendisliği faaliyetlerini gerçekleştirmekte ve kullanım senaryolarını oluşturmaktadır. Çevik proje yönetim aracında, işler için önerilen durumlar (Planlanan, Devam Eden, Alt Yüklenicide Devam Eden, Tamamlanan) ve iş akışları hayata geçirilmiştir.

Çevik yaklaşım önerisi, proje kapsamında hayata geçirilerek iki aylık bir dönem kapsamında uygulanmıştır. Uygulama dönemi içerisinde sayısal kıyaslama gerçekleştirilecek düzeyde veri toplanamamıştır; ancak alt yüklenici yönetimindeki çevik yaklaşımın projeye şu katkıları sağladığı görülmüştür:

- Projedeki işlerin tüm ilgililere görünürlüğü artmıştır.
- Proje geçmişinin ve hafızasının kayıtlı olması sağlanmıştır.
- Alt Yüklenici Sorumlusu tarafından yürütülen işler tanımlı hale gelmiştir.
- Ürün sahipliği faaliyetleri kapsamında projelerin birbirine olan etkisi kayıt altına alınmıştır.
- Kullanım senaryoları daha kısa sürede netleşmiştir.
- Çevik proje yönetim ortamı ile aynı anda birden fazla proje için alt yüklenici yönetimi faaliyetleri gerçekleştirilmiştir.
- Alt Yüklenici tarafından gerçekleştirilen iterasyonlardaki kapsam daha net planlanmıştır.

6 Sonuçlar ve Gelecek İçin Planlar

Alt yüklenici yönetimi için modellenen çevik yaklaşım ASELSAN REHİS bünyesinde birden fazla projeye çıktı veren bir yazılım ürün hattı projesinde hayata geçirilmiştir. Yaklaşımın çevik bir proje yönetimi ortamında hayata geçirilmesi, alt yüklenici yönetimi faaliyetlerinin Alt Yüklenici Sorumlusu kapsamındaki tüm ilgililere görünürlüğünü artırmış ve faaliyetlere ilişkin geçmişin kayıtlı olmasını sağlamıştır.

Ürün sahipliğine ilişkin faaliyetlerin öncelikli olarak Alt Yüklenici Sorumlusu tarafından gerçekleştirilmesi sonucunda kullanım senaryolarının netleşmesi çalışmalarının süresinin kısaldığı görülmektedir. Gelecek dönemde, senaryo çalışmalarındaki süre kısalmasının projenin toplam süresine etkisi ölçülerek değerlendirilecektir. Bu kapsamda senaryo değişikliklerinin de ele alınması faydalı görülmektedir. Doğrulama kapsamında da gelecek dönemde; resmi süreçler öncesinde Alt Yüklenici faaliyetlerinin çevik süreçler kapsamında doğrulanmasının kabul faaliyetlerindeki hata sayısına etkisi değerlendirilecektir.

Kaynakça

1. Standish Grup Ana Sayfası, https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf, Erişim Tarihi 2019/06/20.
2. D. Sağlam, Alt Yüklenici Seçimi, Yönetimi, Yaşanan Sorunlar, Çözüm Süreçleri Ve Ekosistem Geliştirilmesi Hakkında Bir Çalışma, https://www.tskgv.org.tr/contents/makaleler/599, Erişim Tarihi 2019/06/20.
3. C. Gencer, A. Kayacan, "Yazılım Proje Yönetimi: Şelale Modeli ve Çevik Yöntemlerin Karşılaştırılması", Bilişim Teknolojileri Dergisi, 335-352, 2017.
4. L. Thorup and B. Jensen, "Collaborative Agile Contracts", 2009 Agile Conference, Chicago, IL, pp. 195-200, 2009.
5. M. Fowler and J. Highsmith, "The agile manifesto", Softw. Dev., vol. 9, pp. 28–35, 2001.
6. K. Schwaber and J. Sutherland, "The Scrum Guide", November 2017, https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf, Erişim Tarihi 2019/06/20.
7. D.J. Anderson, "Kanban: Successful Evolutionary Change for Your Technology Business", Blue Hole Press, ISBN 9780984521401, 2010.
8. D. I. K. Sjøberg, A. Johnsen and J. Solberg, "Quantifying the Effect of Using Kanban versus Scrum: A Case Study," in IEEE Software, vol. 29, no. 5, pp. 47-53, Sept.-Oct. 2012.
9. M. Stoica, B. Ghilic-Micu, M. Mircea, C. Uscatu, "Analyzing Agile Development – from Waterfall Style to Scrumban", Informatica Economica. 20, pp.5-14, 2016.
10. D. Turk, et al., "Limitations of Agile Software Processes," Proc. The 3rd International Conference on eXtreme Programming and Agile Processes in Software Engineering, Springer-Verlag, pp. 43-46, 2002.
11. M. Paasivaara and C. Lassenius, "Could Global Software Development Benefit from Agile Methods?," 2006 IEEE International Conference on Global Software Engineering (ICGSE'06), Florianopolis, pp. 109-113, 2006.
12. M. Penttinen and T. Mikkonen, "Subcontracting for Scrum Teams: Experiences and Guidelines from a Large Development Organization", 2012 IEEE Seventh International Conference on Global Software Engineering, Porto Alegre, pp. 195-199, 2012.

(De)Motivation Factors in Agile Teams from the Design Perspective

Necmettin Ozkan

Kuveyt Türk Participation Bank, R&D Center, Kocaeli, Turkey
necmettin.ozkan@kuveytturk.com.tr

Abstract. This study aims to reveal the factors that have an impact on employee (de)motivation, through the fundamental design features of agile software development in general and Scrum in particular. To determine the fundamental features of design dimension of the Agile Software Development, the Agile Manifesto and Scrum Guide are based on. In order to identify the factors, an analysis is made over the Agile Software Development Manifesto, Scrum Guide and 16 studies from a literature review within the same context. Then, the (de)motivation factors are justified and enriched with practice by performing semi-structural interviews. Thus, this study presents a comprehensive and outlined reference including the existing studies of the subject and the first attempt in the literature with its approach to handling the subject merely with its design dimension.

Keywords: Agile Software Development, Scrum, Team, Project, People, Motivation, Demotivation.

Çevik Takımlarda Tasarımsal Boyutuyla (De)Motivasyon Faktörleri

Necmettin Özkan

Kuveyt Türk Katılım Bankası, Ar-Ge Merkezi, Kocaeli, Turkiye
necmettin.ozkan@kuveytturk.com.tr

Özet. Bu çalışma, genelde çevik proje yönetimi ve özelde Scrum'ın temel tasarımsal özellikleri üzerinden, çalışan motivasyonuna etkisi olan faktörleri açığa çıkarmayı hedeflemektedir. Çalışma, temel tasarım boyutunu tanımlamak için Çevik Manifesto ve Scrum Kılavuzu'nu baz almıştır. Çevik Yazılım Geliştirme Manifestosu ve Scrum Kılavuzu üzerinden yapılan analiz sonucu ortaya çıkan ve literatür taraması sonucu ulaşılan 16 adet çalışmanın içeriklerinden tasarımsam boyut ile ilgili olduğu düşünülen faktörler, çalışan motivasyonunu etkileyen unsurlar olarak belirlenmiştir. Sonrasında, teori boyutunda belirlenen bu (de)motivasyon faktörlerinin yarı yapısal görüşmeler yapılarak pratik düzlemde doğrulaması ve zenginleştirilmesi yapılmıştır. Bu çalışma, mevcut çalışmaları derleyerek genel ve özet bir referans olma özelliği sunmaktadır. Ayrıca, konuyu sadece tasarım boyutuyla (Çevik Yazılım Geliştirme

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Manifestosu ve Scrum Kılavuzu bazında), bütünleyici olarak ele alması ile literatüre bir ilk olma özelliği taşır.

Anahtar Kelimeler: Çevik, Scrum, Motivasyon, Yazılım Geliştirme, Proje

1. Giriş

Teknoloji, süreç ve insan üçlemesinde insan ayrı bir yere sahiptir. Hem potansiyelinin aralığı esnek ve geniş olabilmekte hem de teknolojiyi ve süreçleri tasarlayıp onlara şekil vermektedir. İçinde bulunduğu kurumu etkileyen önemli faktörlerdendir ve proje başarısını doğrudan etkiler [21]. Her ne kadar insanın efektif yazılım geliştirme süreçleri için önemi bilinse de [1,2,3], insan faktörü göz ardı edilmeye devam edilmekte [4, 12] ve bu ciddi bir sorun olarak kendini göstermektedir [14].

Motivasyon ise üretkenlik, kalite ve proje başarısını direkt etkileyen önemli bir insan unsurudur [18, 19]. Motivasyonu ekonomik girdiler dışında etkileyen birçok etken vardır [16] ve çevik proje yönetimi, çalışan motivasyonunu farklı yönde etkiler [11, 12, 19, 28]. İnsan faktörünün ve motivasyonun önemine rağmen, insanı odağına alan ve son zamanlarda çokça kullanılmaya başlanan çevik projesi yönetimi özelinde çalışan motivasyonunun ele alındığı çalışma sayısı göreceli olarak azdır [12, 25].

Bu çalışma, konuyu tasarım boyutunda ele almayı tercih etmekte ve çevik proje yönetiminin tasarımında (de)motivasyonu etkileyen unsurları irdelemeyi hedeflemektedir. Çalışma, temel tasarım boyutunu tanımlamak için Çevik Manifesto ve Scrum Kılavuzu'nu baz almıştır. Bu kapsamda çalışma iki yol üzerinden teorik düzlemdeki (de)motivasyon faktörlerini açığa çıkarmıştır: Çevik Yazılım Geliştirme Manifestosu [20] ve Scrum Guide [13] üzerinden yapılan analiz sonucu ortaya çıkan faktörler ve literatür taraması sonucu ulaşılan 16 adet çalışmanın içeriklerinden çevik proje yönetiminin kendi evrensel tasarım kurgusu ile ilintili olan faktörler. Bu kapsamda, yazılım geliştirme ve proje yönetimi yöntemlerinden bağımsız olan fiziksel şartlar, sektör baskı vb. ve Çevik Yazılım Geliştirme Manifestosu ve Scrum Guide'ta ifade edilene göre eksik veya kusurlu uygulamalardan kaynaklanan (Scrum Master'ın kontrol odaklı olması, günlük toplantıların hesap vermeye dönüşmesi, müşterinin sürece dahil olmaması, toplantıların eksik yapılması gibi) etkenler göz ardı edilmiştir.

Birincil kaynak olarak saha çalışmalarının seçilmemesinde saha çalışmalarının bağlamının, özellikle içinde insan parametresini barındırıyorsa, çeşitlilik ve genelleme yapamayacak kadar farklılık göstermesidir [12, 19, 28]. Ek olarak saha çalışmalarının bütünü kapsayacak kabiliyette olmaması, ilgili tüm faktörlerin belirlemesini zor kılmaktadır. Bu nedenlerden dolayı sağlıklı bir belirleme için ilk aşamada pratik düzlem yerine öncelikle, bağlamdan bağımsız olarak, teori ve tasarım üzerinden bir incelemeye gidilerek çalışmanın pratik etkilere bağlı bozulmalardan soyutlanması hedeflenmiştir. Tasarım boyutunda (de)motivasyon faktörleri belirlendikten sonra elde edilen çıktının pratik düzlemde sınanarak doğrulaması ve gerekli ise zenginleştirilmesi yapılmıştır. Bunun için, teori boyutunda belirlenen (de)motivasyon faktörleri yarı yapısal görüşmeler yapılarak pratik ile kıyaslanmış, gerekli ise faktörler kümesine eklemeler yapılmıştır.

Çalışma, odağına genel manası ile çevik yaklaşımı almakla birlikte pratikte yaygın olarak kullanılan Scrum çerçevesi özelinde değerlendirmeler yapmıştır. Bu yolla, sahadaki alan genişliğini kapsama ve pratik kullanımla somutlaştırma amacı güdülmüştür.

2. Benzer Çalışmalar

Bu çalışmanın kapsamına giren çalışmalara ulaşmak için yapılan literatür taramasında "agile motivation" anahtar kelimeleri ile 02-03/05/2019 tarihlerinde arama yapılmıştır. Dönen sonuç sayısı inceleme için makul olmadığı durumlarda aramanın yapıldığı yer üzerinden daraltma yapılmıştır. Arama yapılan kütüphane, arama kelimesi, kütüphane içinde arama yapılan içerik türü ve dönen sonuçlar Tablo -1'de paylaşılmıştır. Başlıkta arama yapıldığı zaman, arama yeri daraltıldığı için "Agile" anahtar kelimesine ek olarak "Scrum" anahtar kelimesi ile ayrı bir arama daha yapılmıştır. Konu ile ilintili olma durumunu analiz etmek için "agile satisfaction" kelimeleri ile örnek iki kütüphanede arama yapılmış, dönen sonuçların konunun içine dahil olması durumuna bakılarak bu anahtar kelimelerle daha geniş kapsamda arama yapılmasına gerek görülmemiştir.

Tablo 1. Literatür Taraması

Kütüphane	Arama Kelimesi	Arama Yeri	Dönen Sonuç
ACM	agile motivation	Tüm İçerik	95
DOAJ			17
IEEE			96
Scopus			359
Google Scholar		Başlık	23
Springer			3
Google Scholar	scrum motivation	Başlık	2
Springer			0
DBLP	agile motivation	Metadata	14
WoS	agile motivation	Metadata	223

*WoS ve DBLP tam metin içinde aramaya imkan vermemektedir.

Arama sonuçlarından dönen toplam 832 çalışma başlık ve gerekli olması durumunda özet metinleri üzerinden incelenmiştir. Çalışmaların seçilmesinde aşağıdaki dahil etme (D) ve hariç tutma (H) kriterleri uygulanmıştır:

D1: Hakem değerlendirmesinden geçen,

D2: İngilizce olan,

H1: Yazar tarafından tam metnine erişimi olmayan (3 tane çalışma yazarın tam metne erişimi olmadığı için kapsama alınmamıştır),

H2: Formal olmayan (tez, deneme vb.)

H3: Özelde Scrum dışında bir çevik yönteme (XP, FDD vb.) odaklanan çalışmalar

Seçilen 35 çalışma tam metinleri üzerinden incelenmiş, bu çalışmanın bağlanımda 15 çalışma belirlenmiştir. Aynı yazarlara ait benzer 2 çalışmanın metodunda (anahtar arama

kelimelerinde) hata bulunması nedeniyle bu çalışmalar hariç tutulmuştur. Belirlenen 13 çalışma referans listesi ile birlikte incelenmiş ve ilgili olduğu tespit edilen 3 adet çalışma, [5, 6, 19], kapsama eklenmiştir. 16 adet çalışma içinden bu çalışma ile benzer şekilde çevik proje yönetimi (de)motivasyon faktörlerini kapsamlı bir şekilde inceleme hedefinde olan 7 tane çalışma, [7, 10, 12, 22, 25, 28, 30], tespit edilmiştir. Belirlenen 7 adet çalışmanın listesi ile tüm referans kaynakların içerdikleri (de)motivasyon faktör adetleri (Tablo-2) paralellik göstermekte, ilk beş sıradaki kaynağın kapsandığı görülmektedir. Böylece her iki yolla (hem içerik hem de faktör adedi bazında) benzer bir listenin oluştuğu söylenebilir.

Tablo 2. Referans Kaynakların Faktör Adetleri

Referans Kaynak	Faktör Adedi
12	17
25	16
22	14
30	13
7	12
9	9
8	6
28	6
19	5
29	4
27	4
6	3
10	2
24	2
5	1
23	1
Toplam	***115***

Çalışma [12], çevik bağlamdaki motivasyon faktörlerini 2012 yılına kadar olan sistematik literatür gözden geçirme ve üç farklı şirkette yapılan saha taraması ile ortaya koymaktadır. Çalışma [28], iki çevik takım üzerinden yapılan incelemeler ile çevik projelerde motivasyon faktörlerini incelemektedir. Çalışma [10] yazılım geliştirme test uzmanları açısından çevik ve klasik yöntemlerin motivasyonu nasıl etkilediğini nitel bir anket üzerinden incelemiştir. Çalışma [30], "İlerleme Prensibi" teorisi üzerinden çalışan motivasyonu Scrum pratiklerini nasıl etkilediğini araştırmıştır. [25], üç çevik uygulamanın (günlük stand-up, iterasyon planlama ve iterasyon retrospektifi) çevik takımların motivasyon ve de-motivasyonuna nasıl etkide bulunabileceğini araştırmıştır. Aynı yazarın [22] çalışması benzer bir kapsamı iki ülke özelindeki takımlar ile ortaya koymaktadır. Çalışma [7], çalışan motivasyonunu ve özelde yazılım geliştirici motivasyonunu etkileyen faktörleri ortaya koyduktan sonra çevik yazılım geliştirme yaklaşımının bu faktörlere ne ile ve nasıl katkı sağlayabileceğine dair kişisel görüşleri sunmuştur.

İlgili bu çalışmaların dışında kalan diğer 9 çalışmanın temel odağı çevik yöntemlerde çalışan motivasyonu olmasa da bu çalışmalar içerikleri itibariyle kısmı olarak bu çalışmaya girdi sağlar niteliktedir. 16 çalışmadan gelen bu girdiler Tablo 4'de iletilen referans listesinde aktarılmıştır.

Böylece bu çalışma mevcut çalışmaları derleyerek genel ve özet bir referans olma özelliğindedir. Ayrıca, konuyu sadece tasarım boyutuyla bütünleyici olarak ele alması ile literatürde bir ilk olma özelliği taşır. Bu özelliği ile de evrensel olarak (bağlamdan bağımsız) ve çevik yazılım geliştirme ve proje yönetiminin kendi orijinal tasarımsal kurgusunun çalışan (de)motivasyonuna etkisini ortaya koymaktadır.

3. Metot

Çalışma, (de)motivasyon faktörlerini belirlemede literatür taramasını ve yazar tarafından yapılan analizi yöntem olarak kullanmıştır. Seçilen literatür kaynakları ve kaynaklar içinden ilgili içeriklerin çıkartılması ve analiz edilen Çevik Yazılım Geliştirme Manifestosu ve Scrum Kılavuzu içinden ilgili içeriklerin belirlenmesi, belirlenen tüm bu içeriklerin sınıflandırılması yazar tarafından yapılmıştır. Ardından elde edilen listede, yapılan yarı yapısal görüşmeler üzerinden (analiz seti için) doğrulamaya ve (tüm set için) zenginleştirmeye gidilmiştir. Bu görüşmelerde olası eklemeler için, görüşülen kişi serbest bırakılarak içerik sağlaması hedeflenmiş, yazarın analiz maddelerini doğrulama için ise her madde bazında görüşülen kişiden değerlendirme yapması talep edilmiştir. Literatürün sağladığı mevcut liste için bir doğrulama yapılmamıştır. Buna karşın literatür ile aynı madde görüşülen kişi tarafından da ifade edilmişse buna Tablo-4'te yer verilmiştir.

Yapılan yarı yapısal görüşmeler özelinde çevik yöntemi yeteri zaman deneyimleyen, kurum kültürü çevik yöntem özelinde tam olarak oturmuş, çevik yöntemlerin dönüşüm zorluklarından ziyade, uzun soluklu çalışmalarıyla ve her tür projeyle çalışmış kişilerle örnek vaka çalışması yapılması sağlıklı olacaktır. Aksi durumda, örneğin küçük projelere odaklı çevik uygulamalar ile kritik ve büyük proje geliştirmesinde kullanılan yöntemlerin kıyaslaması denk bir platformda olmayacaktır. Çevik yöntemlerin yeni ve yükselişte olması [27], klasik yöntemlere nazaran son zamanlarda daha çok popüler olması ve klasik yöntemleri dışlayarak bir alternatif olarak kendini sunması çalışanların algılarını bir şekilde etkileyecek ve bu da dengeli bir değerlendirmeyi olumsuz etkileyecektir. Bu nedenle, görüşme yapılan kişilerin bu özellikleri barındırmasına özen gösterilmiştir. Bu kısıtlar gözönüne alınarak seçilen ve görüşme yapılan, Türkiye bankacılık sektöründen, 3 kişinin sektör ve Scrum ile geliştirme deneyimi ve rolleri Tablo 3'te iletilmiştir.

Tablo 3. Görüşülen Kişi Bilgileri

Görüşülen Kod	Toplam Tecrübe	Scrum Tecrübe	Rol	Scrum Deneyim Firma
G1	11	4	Analist	A
G2	14	3	Yazılımcı	A
G3	6	2	Yazılımcı	B

4. Motivasyon Faktörleri

Çevik proje yönetiminde takım içindeki her kişinin kendi aralarındaki ve müşteri ile olan iletişim kanalları ve noktaları çokludur. Geliştirmenin şelale yöntemindeki gibi seri ve sıralı dizilmesi, geliştiricilerin de birbirine iletişim noktalarını bire-çok yerine bire-bir şeklinde

seri ve sıralı bağlanması manasına gelir. Çevik takım yapısı bu tek düzlemi, takımın her bireyini takım çatısı altında bir araya getirerek kırar ve zamanlama olarak bir sıra gözetmeyerek iletişim boyutunu arttırır.

Çevik proje takımlarında, takıma verilen yetkiler ile kişilerin kendi kendini yönetme, sorumluluk, seçme ve inisiyatif alma imkanı [8, 9, 12, 19, 22, 25, 28, 29], sık ve erken geri bildirim alma imkanı [7, 8, 9, 12, 22, 25, 29, 30, 31], birlikte çalışma [7, 9, 12, 19, 22, 24, 25, 27, 28, 30], eşitlik sağlama[12], müşterinin sürece dahil olması [12, 19], ait olma hissi [19, 25], bilgi paylaşım imkanı [9, 12, 22, 28, 30], ortak hedefe odaklanma [7, 22], takımın uçtan uca ürüne hakim olması [12, 19, 29, 30], geliştiricinin odağını koruyabilmesi [22, 30], sağlanan katkının ortaya çıkması, hissedilmesi ve başkaları tarafından görülmesi [7, 12, 22, 25, 29, 30], iletişimin kolay, açık ve hızlı olması [7, 8, 9, 10, 12, 22], yüksek standartta geliştirme imkanı [7, 8, 12], ve takımın üyelerine gelişme ve öğrenme imkanı tanınması [7, 12, 30] çalışan motivasyonunu olumlu etkiler. Takımın hem dışa hem de kendi içindeki değişiklik ihtiyacına cevap verebilme yeteneği [9] çalışanı motive eder [12], adaptasyon yeteneğini arttıran süreçler çalışanın iş-özel hayat dengesi kurmasını kolaylaştırır [12].

Çevik proje takımlarının motive olmayan ekip üyelerine karşı veya organizasyonun motive olmayan takımlara karşı yaptırımları ile ilgili dengeleyici bir yaklaşım sunulmamaktadır. Örneğin kendine konfor alanı bulan takım üyesine veya takıma karşı yaptırım gibi denge unsurları gerekli olabilir. Motive olmayan takım üyesinin olumsuz etkileri [22, 25, 28, 30] ve bu etkinin motive olma durumuna göre üç kat daha fazla etkili olma gücü [30] göz önüne alındığında, bu alanın çevik proje takımlarını ve daha genel manada çevik yöntemleri riske sokması olasıdır. Benzer durum takımın kritik yapma, olumsuz geri bildirimlerde bulunma ve çatışmadan kaçınma durumu için de geçerlidir [25].

Her ne kadar kendine yeten bir yapıda çevik takımlar kurmak bir avantaj olsa da, bu bir zaman sonra takımların kendi içinde homojen kalmasına, dış dünyadan farklılaşarak izole olmasına [7, 30] ve merkezi tasarımlardan ve yapılardan ayrılmalarına neden olacaktır.

Takım içindeki homojen yapıya bağlı olarak, idari pozisyonlardaki basamakları içeren bir hiyerarşiden ziyade düz bir organizasyon yapısı, kurumların kendi sınırları bir yana, insan yönetimi deneyimini değerli gören sektör için bir dezavantaja neden olur [25]. Ayrıca takımın içinde bireysel çabaların dışarıdan görülemeyip kariyer imkanına dönüşmeme riski de vardır [6]. Çalışma [7], tam tersi yönde çevik takımlarda bireysel çabaların dışarıdan daha net görülebildiğini ifade etmektedir. Bununla birlikte çevik yöntemler ile çalışmanın sektörde bir prestij ve buna bağlı olarak bir kariyer avantajı olduğu yönünde görüşler de vardır [28].

Çevik Manifesto içinde değil ama Scrum'da ürünün merkezi konumu belirgindir. Daha büyük perspektif sağlayan hizmet ve proje gibi olgulara bağlı tatminin geri planda olması, ölçeklenebilir uygulamalarda çevik proje yönetiminin sınırlı kalması ve buna bağlı olarak büyük resmi çizememe/görememe durumu çalışanın göreceli olarak büyük çaplı işlere bağlı tatminlerden uzak kalmasına neden olur. G1'e göre Scrum'da biten işler daha çok iç tatmin sağlamakta ve küçük işler takım dışından bir noktadan takdir görmeyi azaltmaktadır. Klasik yöntemlerde başarılan büyük parçanın takdiri daha görünür ve fazla olmaktadır [G1].

Takım aktiviteleri de ürün odağından dolayı geliştirme aktiviteleri etrafında kümelenmiştir. Bu aynı zamanda takım içindeki yük dengesinin, homojen rollere rağmen, bir alt kırılımda salt manadaki ürün geliştirme rolleri eksenine kaymasına ve takım içi dengenin bozulmasına neden olur [9]. Bu denli ürün odağının olduğu sürekli bir akış içinde, takımların kendine alan oluşturarak insana dair değerleri yönetecek dengeyi bulması şarttır [5].

Şelale yönteminde, geliştirme çalışmalarının gerçek değere dönüştüğü noktaya uzaklık fazla iken, çevik yaklaşımda bu mesafe kısadır. Çevik yapılardaki iterasyonlar ve geçişler, işi küçük fakat net ve ulaşılabilir hedeflere dönüştürmeyi sağlar [7, 12, 25, 28, 30] ve çalışanların kendine olan ve müşterilerle aralarındaki güveni arttırır [7, 25]. İterasyonlu ilerleme, ürünün gerçek kullanıma daha sık geçmesinden dolayı değer görme hissini canlı tutar [12]. İterasyonlar, projenin ilk kaynağı olan müşteriye ve fikrin doğma zamanı ve yerine yakınlaştırır. Çevik yapılardaki toplantılar ve iterasyon, ilerleme hissini kuvvetlendirme [8, 9, 12, 25, 27, 30], iletişimi kolaylaştırma ve hızlandırma üzerinden motivasyonu olumlu etkiler [7, 8, 9, 10, 12, 22]. Toplantılar sosyalleşme ihtiyacı olan kişilerin bu ihtiyacını karşılamaları için birincil araçlardan biri olarak da iş görür [15]. G2'ye göre Scrum toplantılarının tanımlı bir mekaniklikte olması nedeniyle bu toplantıların hali hazırda sosyal olan kişilere etkisi düşük, sosyal olmayan kişiler için etkilisi ise olumludur. Diğer yandan yazılım geliştiriciler sık yapılan ve üretken olmayan toplantılara karşı isteksizdirler [22, 30]. Gelişime olan istekleri [17], zamanı efektif kullanmayı gerektirir. Katma değeri düşük toplantılar kendini geliştirmek isteyen yazılım geliştiriciler için zaman kaybı olarak görülür ve motivasyonu olumsuz yönde etkiler [22, 25]. Sadece ritüellere uymak için gerekli olmayan toplantıların yapılması bu durumu pekiştirir. Ayrıca Scrum çerçevesi ile gelen ritüel setinin gerçek hayatın birincil planda olmasının önüne geçmesi bir sorun oluşturur [G2]. Geliştirici gerçek hayattan tam manası ile soyutlanamayacak ve beklenen ritüellerin katı ve esnemeyen planına uyum sağlamakta zorluk yaşayacaktır [G2]. Örneğin gerçek ortam hatası ile ilgilenmesi gereken takımın veya geliştiricinin zamanı net bir şekilde belli olan Sprint planlama toplantısına katılma zorunluluğu, çerçeve ve gerçek ortam realitesi tercihi arasında kalan geliştiricinin çerçeveden kaynaklanarak yaşadığı bir sorundur [G2].

Günlük toplantılar, zaman açısından rutin ve bir öncekine benzeme açısından homojen olmaya yatkındır [22, 25]. Sürekli kısa sprintler koşmak (run), geliştirici açısından bir zaman sonra baskı [9, 10, 27] ve buna bağlı olarak stres üretebilmekte [9, 12, 22, 25], rutine dönme durumu olabilmekte [8] ve bu rutin çalışanı sıkabilmektedir [6].

Günlük yapılan toplantıda, bilinç altı olarak, sürekli ilerleme dayatıldığı gibi, retrospektif toplantılarında sürekli iyileştirme dayatılmaktadır [22, 25]. Geliştiricinin sürekli akan rutinin spesifik bir örneğinde ilerlememe veya kendini iyileştirmemeyi seçmesi teorik olarak mümkün olsa da pratikte bunun tercih etmesi zordur. Kendi çözümü olarak takımı yanıltma yollarına girebilir [25] ve bu da takımdaki güveni zedeler. Bu denli sık rutin içinde geliştiricinin kendine bir alan ve dinlenme zamanı bulması zordur. Sürekli sprintler koşmak (run) insan motivasyondaki inişli çıkışlı tabi dalgalanmalara imkan tanımazken, çalışanda bir zaman sonra yorgunluk üretir. Üretim sektörü, klasik yöntemleri nasıl şekillendirdi ise, Scrum da bu yaklaşımdan etkilenmiş, ağır planlama dışlansa da, ileriye dönük zamanın önden sabitlenmesi (iterasyonlarda olduğu gibi) ve etkinliklere ait maksimum sürenin sabit olması ile planlama olgusu ağır olmasa da daha keskin boyutuyla yaşamaya devam etmiştir. G2'ye göre dinamizmi yüksek genç geliştiriciler için bu denli net bir son tarihin motivasyonu arttırması mümkün iken ilerleyen yaşa bağlı olarak geliştiricilerin bu denli sert ve net bir son tarihin baskısını kabullenmesi kolay olmayacaktır. Yine de Sprint süresinin sabit olması ile oluşan son tarih parametresi, [23, 30]'e göre çalışan motivasyonunu arttırmaktadır. Bu sonuç odaklı bakış açısı, Sprint ve genel olarak geliştirme boyunca fazlalıkların elimine edilmesi sonucu ile çalışanı olumlu yönde etkiler [7, 12].

Çevik yöntemler, basit ve pratik dokümantasyon yaklaşım ile çalışanı motive eder [24]. Çevik Manifesto içinde kapsamlı dokümantasyondan ziyade çalışan yazılımın yeğlendiği

vurgulanır [20]. İnsan doğasının insan-insan yolunu iletişim kanalı olarak tercih etmesinden dolayı, yüz yüze iletişimin doküman üzerinden iletişime tercih edilmesi makuldür [G1]. Fakat dokümantasyon gereğinden fazla veya eksik yapılması ile bir de-motivasyon faktörüdür. Eksik yapılması durumunda kurumsal hafızada neden olduğu zayıflama orta vadede çalışan aleyhine olmaktadır [G1, G3].

Çevik yöntemlerde dokümantasyon konusu bu kapsamda tasarım ve iletişim bağlamında yeniden şekillendirmeye çalışırken, genel manadaki dokümantasyon için tanımsız bir alan bırakmıştır. Çalışanın destekçisi olan dokümantasyon, araç ve süreçler ve beraberinde getirdiği dijitalleşme yeteneğinin çevik yöntemlerde geri planda kalması insan tarafında genel manada yetkinlik eksikliği oluşturma riskini barındırır. Bu konuda organik denge olmalı, takımlar ve kurumlar ihtiyaç olan araç ve süreç seviyesini çerçevelere göre değil, kendilerine göre belirlemelidir [G2].

Çalışan boyutunda, yazılımı, geç aşamada da gelse değişimi, müşterinin menfaatini ön plana çıkaran çevik yöntemler, genel manada müşterinin hızlı kazancı için geliştiriciyi feda etme riski barındırır. Çevik yöntemlerdeki, dokümantasyon yaklaşımlarında olduğu gibi, kısa vadedeki kazanımlar için geliştiricinin orta ve uzun vadede olumsuz etkilenmesi ve de-motive olması riski vardır. Örneğin, geliştiriciyi koruyan tasarım doküman onayları çevik yaklaşımda tavsiye edilmemektedir [G1].

Belirlenen bu faktörler insan odağı, iterasyon ve toplantılar, süreç ve iletişim kategorileri altında ilgili referansları ile birlikte aşağıdaki gibi özetlenerek listelenmiştir. Tablonun "Motivatör/De-motivatör "olarak sınıflandırdığı maddeler, ilgili faktöre hem motivasyon hem de de-motivasyon unsuru olarak bakan farklı kaynakların veya analizin olduğunu ifade etmektedir. Yazar tarafından yapılan analiz literatürle örtüştüğü takdirde, literatür baskın olarak kabul edilmiş ve mükerrerliği önlemek için sadece ilgili literatür baz alınmıştır. Yazarın analizi sonucu ortaya atılan görüş literatürden farklı ise "Analiz" ifadesi ile ilgili maddeye ekleme yapılmıştır.

Tablo 4. (De)Motivasyon Faktörleri

Kategori	#	Boyut	Faktör	Literatür Referans Adedi	Referans
Motivatör	1	İletişim	Bilgi paylaşım imkanı	5	[9, 12, 22, 28, 30]
	2	İletişim	İletişimin kolay, açık ve hızlı olması	6	[7, 8, 9, 10, 12, 22], G3
	3	İnsan Odağı	Ait olma hissi	2	[19, 25]
	4	İnsan Odağı	Birlikte çalışma	10	[7, 9, 12, 19, 22, 24, 25, 27, 28, 30], G3
	5	İnsan Odağı	Çalışanların kendine olan ve müşterilerle aralarındaki güvenin artması	2	[7, 25]
	6	İnsan Odağı	Gelişme ve öğrenme imkanı	3	[7, 12, 30]
	7	İnsan Odağı	Ortak hedefe odaklanma	2	[7, 22]
	8	İnsan Odağı	Kendi kendini yönetme, sorumluluk, seçme ve inisiyatif alma imkanı	8	[8, 9, 12, 19, 22, 25, 28, 29], G3
	9	İnsan Odağı	Sosyalleşme imkanı	0	Analiz, G2

	10	İnsan Odağı	İş-özel hayat dengesi	1	[12]
	11	İnsan Odağı	Eşitlik	1	[12]
	12	Süreç	Geliştiricinin odağını koruyabilmesi	2	[22, 30]
	13	Süreç	İlerleme hissi	6	[8, 9, 12, 25, 27, 30]
	14	Süreç	İşi net ve ulaşılabilir hedeflere dönüşmesi	5	7, 12, 25, 28, 30]
	15	Süreç	Müşterinin sürece dahil olması	2	[12, 19]
	16	Süreç	Sağlanan katkının ortaya çıkması, hissedilmesi ve başkaları tarafından görülmesi	6	[7, 12, 22, 25, 29, 30]
	17	Süreç	Sık ve erken geri bildirim alma imkanı	9	[7, 8, 9, 12, 22, 25, 27, 29, 30], G1
	18	Süreç	Uçtan uca ürüne hakim olma imkanı	4	[12, 19, 29, 30], G3
	20	Süreç	Yüksek standartta geliştirme	3	[7, 8, 12]
	21	Süreç	Fazlalıkların elimine edilmesi	2	[7, 12]
	22	Süreç	Değişiklik ihtiyacına cevap verebilme yeteneği	1	[12]
	23	İnsan Odağı	Kariyer imkanı	3	[6, 25, 28]
	30	İnsan Odağı	Takımın içinde bireysel çabaların dışarıdan görülmesi	2	[6, 7]
Motivatör/De-motivatör	24	İterasyon ve toplantılar	İleriye dönük zamanın (sprint süresinin) önden sabitlenmesi	2	[23, 30], Analiz, G2
	25	İterasyon ve toplantılar	Sık toplantılar	2	[22, 30]
	19	Süreç	Dokümantasyon yaklaşımı	1	[24], Analiz, G1, G3
	26	İletişim	Dijitalleşmenin verimli ve etkin kullanılmasının geri planda kalması	0	Analiz, G2
	27	İnsan Odağı	Kritik yapma, olumsuz geri bildirimlerde bulunma ve çatışmadan kaçınma	1	[25], G3
	28	İnsan Odağı	Motive olmayan takım üyesinin olumsuz etkileri	4	[22, 25, 28, 30], G3
De-motivatör	29	İnsan Odağı	Olumsuz durumların takım üyeleri tarafından gizlenmesi (takımı yanıltma)	1	[25], G1
	31	İterasyon ve toplantılar	Katma değeri düşük toplantılar	2	[22, 25], G1
	32	İterasyon ve toplantılar	Ritüellere uymak için gerekli olmayan toplantıların yapılması	0	Analiz, G1, G2
	33	İterasyon ve toplantılar	Ritüellerin rutine dönmesi ve sıkıcı olması	4	[6, 8, 22, 25], G1
	34	İterasyon ve toplantılar	Timeboxes (Maksimum sürenin sabit olması)	0	Analiz, G2

	35	Süreç	Müşterinin kazancı için geliştiricinin feda edilmesi	0	Analiz, G1
	36	Süreç	Sıkı rutin içinde çalışanın bir alan ve dinlenme zamanı bulmada zorluk yaşanması	0	Analiz, G1, G2
	37	Süreç	Sürekli (kısa) sprintler koşmanın (run) baskı ve stres üretmesi	6	[9, 10, 12, 22, 25, 27], G1, G2, G3
	38	Süreç	Sürekli ilerleme ve/veya kendini iyileştirme baskısı	2	[22, 25], G1
	39	Süreç	Sürekli sprintler koşmanın (run) yorgunluk üretmesi	0	Analiz, G2, G3
	40	Süreç	Takım aktivitelerinin ürün odağından dolayı geliştirme aktiviteleri etrafında kümelenmesi	1	[9]
	41	Süreç	Takım aktivitelerinin ürün odağından dolayı insana dair değerleri yönetecek alan olmaması	1	[5]
	42	Süreç	Takım içindeki yük dengesinin homojen olmaması	1	[9]
	43	Süreç	Takımların kendi içinde homojen kalması, dış dünyadan farklılaşarak izole olması, merkezi tasarımlardan ve yapılardan ayrılması	2	[7, 30]
	44	Süreç	Büyük çaplı işlere bağlı tatmin eksikliği	0	Analiz, G1, G2
	45	Süreç	Motivasyondaki inişli çıkışlı tabi dalgalanmalara imkan tanımaması	0	Analiz, G1, G2, G3

* G1: Görüşülen 1, G2: Görüşülen 2, * G3: Görüşülen 3
**Motivatör/De-motivatör kategorisinde altı çizili olan referans de-motivatör faktörüdür.

5. Değerlendirme

Tablo 4'te belirlenen (de)motivasyon faktörleri incelendiğinde yazar tarafından tespit edilen "Analiz" maddelerinin görüşme yapılan kişilerin değerlendirmeleri tarafından desteklendiği görülmektedir. Bazı maddeler için bu destekleme görüşülen kişilerin sadece biri tarafından sağlandığı için bu maddeler özelinde daha fazla araştırmaya ihtiyaç vardır.

Nicel olarak bakıldığında çevik yöntemlerin çalışan motivasyonu açısından klasik yöntemlerden daha avantajlı olduğu görülür [11, 25]. Kendi içinde değerlendirildiğinde bu çalışma içinde yürütülen literatür çalışmasından elde edilen verilere göre %65 oranında motive edici, %35 oranında de-motive edici faktör olduğu görülmektedir. Nihai liste üzerinden bu oran, %52 motive edici, %48 de-motive edici faktör olarak dağılmaktadır. Bu durumda göstermektedir ki mutlak manada çevik yöntemlerin çalışan motivasyonunu desteklediği yönünde bir çıkarıma gidilmemelidir. Her ne kadar mevcut literatür ağırlıkta motivasyon faktörlerine odaklansa da de-motivasyon faktörlerini içeren çalışmalar mevcuttur. Bu çalışmalar kayda değer sayıda ve özellikte de-motivasyon faktörü olduğunu göstermektedir. Buna ek olarak literatürde henüz yer bulmamış fakat çevik proje

yönetiminin tasarımsal olarak içinde bulunan de-motivasyon faktörleri olduğu bu çalışma tarafından ileri sürülmektedir. G2'ye göre Scrum kısmi olsa da tasarımsal boyutta gerçeklikten kopuk, kusurlu ve gerçekte uygulanması zor bir önerme ile gelmekte ve bu durum çalışan tarafında zorlanmaya, yıpranmaya, de-motive edici etkiler yaşamaya neden olmaktadır. Bu tip de-motivasyon faktörlerinin literatüre henüz girmemesinde mevcut literatür çalışmalarının çoğunlukla saha çalışmalarından beslenmesinin ve spesifik bir saha örneğinin tüm faktörleri kapsamasının zor olması rol oynamış olabilir. Örneğin çevik proje yönetimine yeni geçiş yapmış takımlarda ritüellere dair yorgunluğun oluşması veya büyük ölçekte proje geliştirmesi yapmayan bir takımda/kurumunda çevik proje yönetiminin konfor alanı dışından bir zorluğun yaşanması beklenemez.

Bu çalışma (de-motivasyondan ziyade) motivasyon faktörleri üzerine mevcut bir odağın bulunması ve motivasyon faktörlerinin genel manada bilinir olmasına karşın özellikle Scrum'ın tasarım olarak içinde barındırdığı motivasyonu olumsuz etkileyecek risklerin çalışanları etkileyebileceğini ileri sürmektedir. Scrum özelinde, bu bağlamda özellikle müşteri menfaatine hizmet eden agresif bir ürün odağı kurgusunda takımın, takım içinde bireyin kendine ihtiyaç olan alanı bulması gerekir.

Çevik yöntemler ile ilgili olumlu çıkarımlarda, çevik yöntemlerin yeni olmasının [27], kabul gördüğü çalışma türlerinin ve ölçeklerinin etkisi de vardır. Ölçeklenebilir projeler için yaşanan zorluklar farklılık gösterecek ve küçük ölçekte sıkıntı oluşturmayan yeni hususlar ortaya çıkacaktır [25, 26]. Ölçekleme konusu çevik yazılım geliştirme dünyasının önünde hala aktif bir sorun iken [26], çevik proje yöntemini bu faktörden bağımsız bir platformda ele almak sağlıklı olmayacaktır. Diğer bir husus ise, kullanım süreleridir. Çevik proje yöntemlerinin bu çalışma kapsamında saha tarafından sağlıklı olarak değerlendirilmesi için zaman olarak bir eşiği geçmesine ihtiyaç vardır. Ancak bundan sonra yapılacak saha çalışmaları sağlıklı sonuçlar verecektir. Bu nedenle bu çalışma, yapılan mevcut çalışmaları dikkate almakla birlikte farklı bir yol izleyerek, pratikten değil, teori boyutundan ilerlemiştir. Yapılan mevcut çalışmalar, ilgili yerlerde kullanılarak çalışmanın bir tür literatür referans kaynağı özelliği kazanması da sağlanmıştır.

6. Kısıtlar ve Gelecek Çalışmalar

Bu çalışma bazı kısıtlar içermektedir. Literatür taramasında seçilen ilgili kaynakların seçilme metodu ve seçilen kaynaklar içinden ilgili içeriklerin belirlenmesi ve sınıflandırılması üzerinden izlenen yöntem içinde kalıtsal bir sübjektiflik barınmaktadır. Benzer durum yazar tarafından çevik proje yönetimi üzerine yapılan analiz için de geçerlidir. Bu kısıtlar, elde edilen faktör setinin doğruluğuna ve tamlığına bir tehdittir. Doğrulama ve gerekli zenginleştirme için yapılan yarı yapısal görüşmelerin adedi göreceli olarak azdır. Bu görüşmelerde olası eklemeler için görüşülen kişi serbest bırakılarak içerik sağlaması hedeflenmiş olsa da doğrulama için yazarın yürüttüğü analiz sonucu oluşan faktör seti görüşülen kişiye sunulmuş, literatürün sağladığı mevcut liste için bir doğrulama yapılmamıştır. Bu yolun varsayımı literatürden elde edilen faktör setinin doğru olduğudur.

Genel olarak, bu çalışmanın elde ettiği (de)motivasyon faktörleri, etkileri ve etki dereceleri, daha kapsamlı bir saha çalışması ile sınanmak durumundadır. Seçilen örneklem kümesinin, çevik yöntemleri yeteri süre ile deneyimlemiş olması önemlidir. Genişletişmiş pratik çalışma sonrası bu çalışmada elde edilen bulgular gerekli ise revize edilmelidir. Bu ise gelecek çalışma konusudur.

Kaynakça

1. Hall, T. and Wilson, D. Views of Software Quality: A Field Report, lEE Proceedings on Software Engineering, April, 111-118 (1997).
2. Wilson, D., Hall, T. and Baddoo, N. : A Framework for Evaluation and Prediction of Software Process Improvement Success. Journal of Systems & Software 59(2), 135-142 (2001).
3. Hammock, N.: Critical Success Factors in Software Process Improvement, 4th Annual European Process Group Conference, June 6-10 (1999).
4. McDermid, A. and Bennett, K. H.: Software Engineering Research: A Critical Appraisal, lEE Proceedings on Software, 146 (4) August, 179 -186 (1999).
5. Hoda, R.: Self-organizing agile teams: A grounded theory. Victoria University of Wellington (2011).
6. Beecham, S., et al.: Does the XP environment meet the motivational needs of the software developer? An empirical study. Agile Conference (AGILE), (2007).
7. Asproni, G.: Motivation, teamwork, and agile development. Agile Times 4(1), 8-15 (2004).
8. Šteinberga, L., and Darja Š.: Towards a contemporary understanding of motivation in distributed software projects: solution proposal. Scientific Papers, University of Latvia (2011)
9. Whitworth, E., and Biddle, R.: The social nature of agile teams. Agile conference (AGILE), (2007).
10. Deak, A.: A comparative study of testers' motivation in traditional and agile software development. International Conference on Product-Focused Software Process Improvement, (2014).
11. Tripp, J. F., Riemenschneider, C., & Thatcher, J. B.: Job satisfaction in agile development teams: Agile development as work redesign. Journal of the Association for Information Systems, 17(4), 267 (2016).
12. Melo, C.O., Santana, C., Kon, F.: Developers motivation in agile teams. 38th Euromicro Conference on Software Engineering and Advanced Applications, (2012).
13. Sutherland, J., & Schwaber, K.: The Scrum guide. the definitive guide to Scrum: The rules of the game. ScrumGuides. com., (2017)
14. Sharp, H. C., Woodman, M., Hovedon, F. and Robinson, H.: The Role of 'Culture' in Successful Software Process Improvement. 25th EUROMICRO Conference, 170-176 (1999).
15. Khalil, O. E. M., Zawacki, R. A., Zawacki, P. A. and Selim, A.: What Motivates Egyptian IS Managers and Personnel: Some Preliminary Results. SIGCPR97, (1997).
16. Mayo, E.: The Human Problems of an Industrial Civilization, Macmillan (1933).
17. Herzberg, F. I.: Work and the nature of man. (1966).
18. Verner, J.M., Babar, M.A., Cerpa, N., Hall, T., Beecham, S.: Factors that motivate software engineering teams: a four country empirical study. J. Syst. Softw. 92, 115–127 (2014).
19. Beecham, S., Baddoo, N., Hall, T., Robinson, H., Sharp, H.: Motivation in software engineering: a systematic literature review. Inf. Softw. Tech. 50, 860–878, (2008)
20. Beck, K., et al.: The agile manifesto. (2001).
21. Cooke-Davies, T.: The "real" success factors on projects. International journal of project management, 20(3), 185-190 (2002)
22. McHugh, O., Conboy, K., & Lang, M.: Motivating Agile teams: A case study of teams in Ireland and Sweden. In International Research Workshop on IT Project Management, (2010).
23. Harris, M. L., Collins, R. W., & Hevner, A. R.: Agile methods: Fast-paced, but how fast?. AMCIS 2009 Proceedings, 149, (2009).
24. Altameem, E. A.: Impact of Agile methodology on software development. Computer and Information Science, 8(2), 9 (2015).
25. McHugh, O., Conboy, K. and Lang, M.: Using agile practices to influence motivation within it project teams. (2011).
26. Dingsøyr, T. and Moe, N. B.: Research Challenges in Large-Scale Agile Software Development, ACM Software Engineering Notes, 38-39 (2013)
27. Whitworth, E., and Biddle, R.: Motivation and cohesion in agile teams. International Conference on Extreme Programming and Agile Processes in Software Engineering. Springer, Berlin, Heidelberg (2007).
28. Albuquerque, R., et al.: Motivating Factors in Agile and Traditional Software Development Methods: A Comparative Study. Brazilian Workshop on Agile Methods. Springer, Cham (2016).
29. Tessem, B., & Maurer, F.: Job satisfaction and motivation in a large agile team. International Conference on Extreme Programming and Agile Processes in Software Engi1neering. 54-61, Springer, Berlin, Heidelberg 2007.
30. Jansson, T.: Agile Project Management: Is Motivation Theory the Missing Link?. In Modern Techniques for Successful IT Project Management, 138-167, IGI Global, 2015.

ModelDOC: Entegre ve Katılımcı Elektronik Doküman Yönetim Sistemi

Osman Abul[1], Davut Gökhan Karatepe[2], Fadime Şeyda Demetoğlu[2] ve Naim Seyrek[2]

[1] TOBB Ekonomi ve Teknoloji Üniversitesi, Ankara
[2] ZEG Teknoloji, Bilkent Cyberpark, Ankara
osmanabul@etu.edu.tr

Özet. Klasik bakış açısına göre doküman insan okuyucu için hazırlanmış yapısal bir bilgi bütünüdür. Öte yandan modern teknolojik bakış açısı ile doküman ağdaki farklı kaynaklardan çeşitli tipteki bilgileri ilişkilendiren ve ayrıca birbiri ile de ilişkili bilgi grubu olarak tanımlanabilir. Ayrıca doküman oluşturmak çoğu zaman farklı birim ve uzmanlıktan kullanıcıların katılımı ve işbirliğini gerektirir. Mevcut doküman yönetim sistemleri çoğunlukla klasik bakış açısını takip ederler. Bu çalışmada modern teknolojik bakış açısı ile geliştirmiş olduğumuz ModelDOC adlı Elektronik Doküman/Belge Yönetim Sistemi (EBYS) tanıtılmaktadır. ModelDOC'un ana unsuru (i) farklı bilgi sistemleri arasında entegrasyon ve (ii) kullanıcılar arası katılımcılığı sağlamasıdır. Bir doküman model-view mantığında ilişkili olduğu tüm bilgileri tutmakta fakat bunlardan bazılarını bir şablon üzerinden insan okuyucuya göstermektedir. Bu sayede dokümanda insan okuyucunun görmediği bilgiler üzerinden indeksleme ve sorgulama imkanı sağlanmaktadır. ModelDOC'un öne çıkan diğer bir özelliği ise sadece modelleme (kodlama-derleme gerektirmeden) yaparak çekirdek EBYS'nin farklı kurumlar için çevrimiçi özelleştirilebilmesidir. Böylelikle form ve şablon tasarımları, raporlama, indeksleme, arama ve sorgulama dinamik olarak yapılabilmektedir.

Anahtar kelimeler: Yazılım Mühendisliği, Yazılım Entegrasyonu, Doküman Yönetim Sistemleri.

ModelDOC: An Integrated and Collaborative Electronic Document Management System

Osman Abul[1], Davut Gökhan Karatepe[2], Fadime Şeyda Demetoğlu[2] and Naim Seyrek[2]

[1] TOBB Ekonomi ve Teknoloji Üniversitesi, Ankara
[2] ZEG Teknoloji, Bilkent Cyberpark, Ankara
osmanabul@etu.edu.tr

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Abstract. From the classical point of view, the document is a set of structured information for the human reader. On the other hand, from a modern technological point of view, the document can be defined as a group of information that associates various types of information linked together from different sources in the network. In addition, creating documents often requires the participation and collaboration of users from different units and expertise. Existing document management systems often follow the classical perspective. In this study, an Electronic Document Management System (EDMS), which is developed with a modern technological perspective, is introduced. The key elements of ModelDOC are enabling (i) integration between different information systems and (ii) inter-user participation. A document holds all the relevant information in the model-view logic, but shows some of these to the human reader through a template. In this way, indexing and querying is provided on the information that is not seen by the human reader. Another feature of ModelDOC is the ability to customize the core EDMS for different organizations by modeling (without coding-compiling) in an online manner. Thus, form and template designs, reporting, indexing, searching and querying can be done dynamically.

Keywords: Software Engineering, Software Integration, Document Management Systems.

1 Giriş

Klasik bakış açısına göre doküman insan okuyucu için hazırlanmış yapısal bir bilgi bütünü olarak tanımlanabilir [1]. Fakat günümüzde daha çok elektronik dokümanlardan bahsedildiğinden tanım da anlamsal kaymaya uğramaktadır. Öyle ki modern teknolojik bakış açısı ile doküman ağdaki farklı kaynaklardan çeşitli tipteki bilgileri ilişkilendiren ve ayrıca birbiri ile de ilişkili bilgi grubu olarak tanımlanabilir [2].

Doküman Yönetim Sistemi bir kamu kuruluşu için en önemli IT (*Information Technology*) sistemidir [1]. Şöyle ki, Doküman Yönetim Sistemleri bir kurum içindeki çok sayıda ve çeşitlilikteki dokümanların etkin bir şekilde oluşturulması, saklanması, indekslenmesi, aranması ve kullanıcılar arasında sirkülasyonu işlerini yerine getirir. Yönetilen dokümanlar arasında dilekçeler, dilekçelere yanıtlar, faturalar, kurum içi formlar, yönetmelikler, harici ve kurum içi yazışmalar, görevlendirmeler, teknik dokümanlar, standartlar, toplantı notları, emirler gibi çok çeşitli evraklar vardır.

Ülkemizde kurumların EBYS (Elektronik Belge Yönetim Sistemi) ihtiyaçlarını düzenlemek ve bir standartlaştırma getirmek için "TS 13298-Elektronik Belge Yönetimi standardı" [3] getirilmiştir. Bu standart ağırlıklı olarak doküman/belgenin içeriğinden çok şekilsel olarak (üst/alt bilgi, gizlilik, format, imza vb.) gereksinimlerini belirlemeye yöneliktir ve bir EBYS için olması gereken minimumları ifade eder.

Günümüzde çok sayıda kamu kurumu Elektronik Doküman/Belge Yönetim Sistemine geçmiş bulunmaktadır. Fakat bu sistemler halihazırda basılı belgenin elektronik kopyasının yönetimi boyutunda kalmaktadır. Yine de mevcut durumda bir EBYS kuruluşlara maliyet, zaman tasarrufu, kolay ve hızlı yönetim/arama ve şeffaflık avantajları sunmaktadır. Bilgi sistemleri arasında entegrasyonu vurgulayan modern teknolojik bakış açısı günümüzde daha anlamlı olmasına rağmen mevcut doküman yönetim sistemleri çoğunlukla klasik bakış açısını takip ederler.

Bir EBYS tasarımında modern teknolojik bakış açısı doğru olmasına rağmen eksiktir. Çünkü bir doküman farklı kaynaklardan bilgiler içeren bir bütün olmanın yanında, oluşturulması, düzenlenmesi, onaylanması, saklanması, sorgulanması gibi işlemler nedeniyle çok sayıda kullanıcının üzerinde katılımcı olarak iş birliği (*collaboration*) yaptığı bir üründür.

Gerek IT varlıklarının çeşitliliği gerekse iş süreçlerinin aynı olmaması gibi nedenlerle kurumların veri/bilgi entegrasyon ve EBYS ihtiyaçları farklılık arzetmektedir. Hatta aynı sektördeki iki kurumun EBYS ihtiyaçları bile birbirinden farklıdır. Daha da ötesi aynı kurumun EBYS ihtiyaçları zaman içinde değişebilmektedir. Bu yüzden, EBYS ürünlerinin kurum bazında kolayca özelleştirilebilmesi çok önemlidir. Çevrimiçi kullanımda yeni sürümün derleme-kodlama yapılmadan sadece veri/süreç modellemesi ile yapılabilmesi bu kolaylığın önemli bir ölçüsüdür. Türkiye'de EBYS kullanıcıları arasında yapılan geniş çaplı bir anket bu ihtiyacı açıkça ortaya koymaktadır [8].

Bu dokümanda ModelDOC adını verdiğimiz kurumsal veri/bilgi entegrasyonu tabanlı katılımcı bir EBYS ürününün tasarımı ve geliştirimi anlatılmaktadır. ModelDOC'un öne çıkan özelikleri ve avantajları aşağıda listelenmiştir.

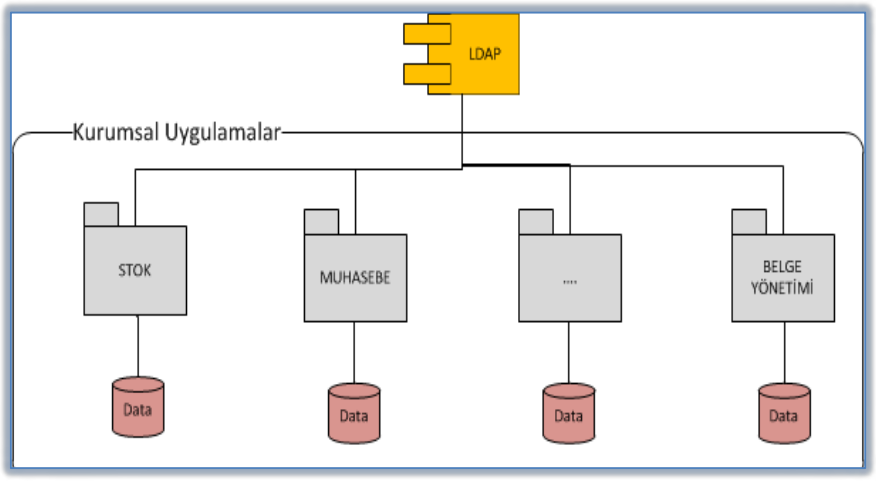
- Kodlama-derleme gerektirmeden sadece veri modellemesi ve şablon tasarımı yapılarak farklı kurumlar için özelleştirilebilme
- Kurumun kullandığı kurumsal uygulamalar ile model seviyesinde entegrasyon
- Kurumun kullandığı uygulamalarda bulunmayan ama ihtiyaç duyulan bilgileri tamamlayabilmek için dahili şema oluşturma
- *Model-view* ayrımı tasarım yaklaşımı [7] ile doküman modeli ve şablon görünümünün ayrı ayrı modellenebilmesi
- CWM (*Common Warehouse Metamodel*) [4] standardını destekleme
- Dağıtık dosya mimarisi ve büyük veri yaklaşımı ile güvenli, güvenilir ve ölçeklenebilir olma
- Çevrimiçi kullanımda dinanik veri modelini destekleme
- Apache Solr [5] ve Elasticsearch [6] alternatifleri ile şemasız dinamik veri indeksleme ve arama
- Belge merkezli veri ambarı sayesinde ile İş Zekası analizleri yapılabilmesi

2 Kurumsal Katılımcı EBYS Entegrasyonu

IT sistemlerinin entegrasyonu temelde uygulama ve servis/veri seviyesinde olmak üzere iki farklı şekilde anlaşılabilmektedir [1]. İlki entegre sistemin ortak bir uygulama da birleştirilmesini ve ikincisi ise, ayrı ayrı bağımsız uygulamaların veri/servis paylaşımını ifade eder.

Herhangi bir büyük ölçekli kurumda EBYS yanında Muhasebe, Stok, İnsan Kaynakları, Finans vb. çok sayıda IT sistemi (kurumsal uygulama) vardır. Bunların geleneksel entegrasyonu Şekil 1'de gösterildiği gibidir. Bu yaklaşımda sistemler tamamen birbirinden bağımsız çalışmakta yalnızca kullanıcılar ve yetkilendirme ortak bir servisten (LDAP) sağlanmaktadır. Bu geleneksel anlamdaki entegrasyon daha çok veri/servis entegrasyonu olmayıp kullanıcı yönetimini (single sign on vb.) kolaylaştırmaya yöneliktir.

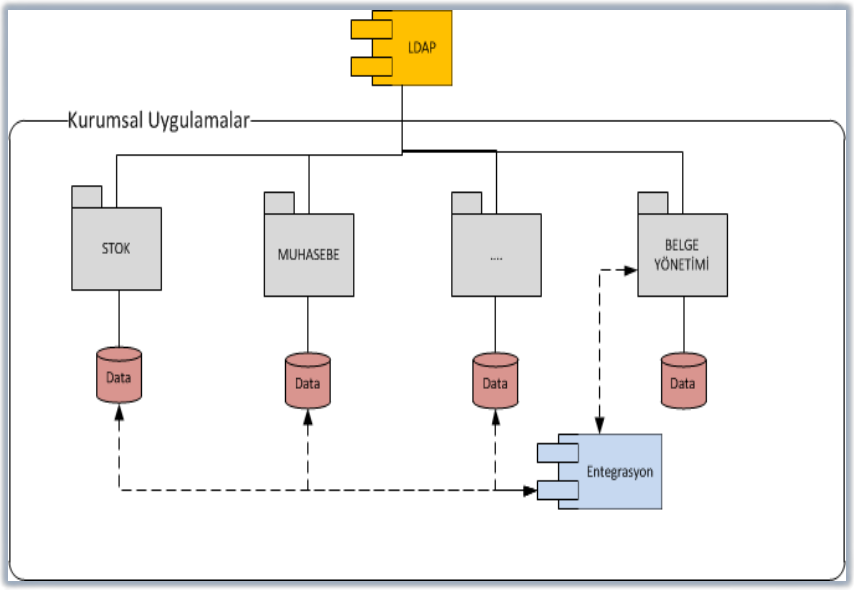
Şöyle bir senaryo düşünelim. Hem Stok hem de Muhasebe sisteminden veri kullanan bir evrak/belge hazırlanması gereksin. Bu durumda, evrakı hazırlayan kişi bu iki sisteme bağlanarak oradan okuduğu (eğer yetkisi yoksa telefonla yetkiliden okuttuğu) verileri manuel olarak bu belgeye girecektir. Burada ortaya temelde iki sorun çıkmaktadır: (i) gerçek anlamda birimlerin kullanıcıları arasında doküman üzerinde doğrudan bir iş birliği olmaması, (ii) evrakın gerçekte entegre olmaması. İkinci sorunu biraz daha açacak olursak, evraktaki bir bilgi kaynak uygulamadakinden kopuk olmakta ve o kaynağın herhangi bir özniteliği ile ilişkilendirilmemektedir.



Şekil 1. Geleneksel anlamda kurumsal EBYS entegrasyonu.

Dokümanların hızlı erişim için indekslenip sorgulandıklarını kabul edelim. Geleneksel entegrasyon yaklaşımında doküman içinde olmayan hiçbir alan indekslenemeyeceği için doküman içeriğinde yer almayan bir öznitelik üzerinden sorgulanamaz. Örneğin, belge içinde sadece stok adı ve miktarı yazılı ise bu belgeyi

ilgili stok kaleminin başka bir özniteliği (mesela türü, sorumlusu vb.) üzerinden sorgulayamayız. Oysa şöyle bir sorgu EBYS içinde ihtiyaç duyulabilir: X kullanıcısının sorumlu olduğu stoklarla ilgili yapılan yazışmaları getir. Benzer şekilde başka bir örnek verecek olursak, sadece öğrenci no ve adının yazılı olduğu bir yurt izin belgesinde, Y üniversitesinde okuyan öğrencilerin izin belgeleri sorgulanamaz. Oysa belge üzerinde yazmasa bile bu bilgiler entegrasyon sayesinde belge ile ilişkilendirilip indekslenebilir ve akabinde sorgulanabilir durumdadır.



Şekil 2. Veri/servis seviyesinde kurumsal EBYS entegrasyonu.

3 ModelDOC Tasarımı

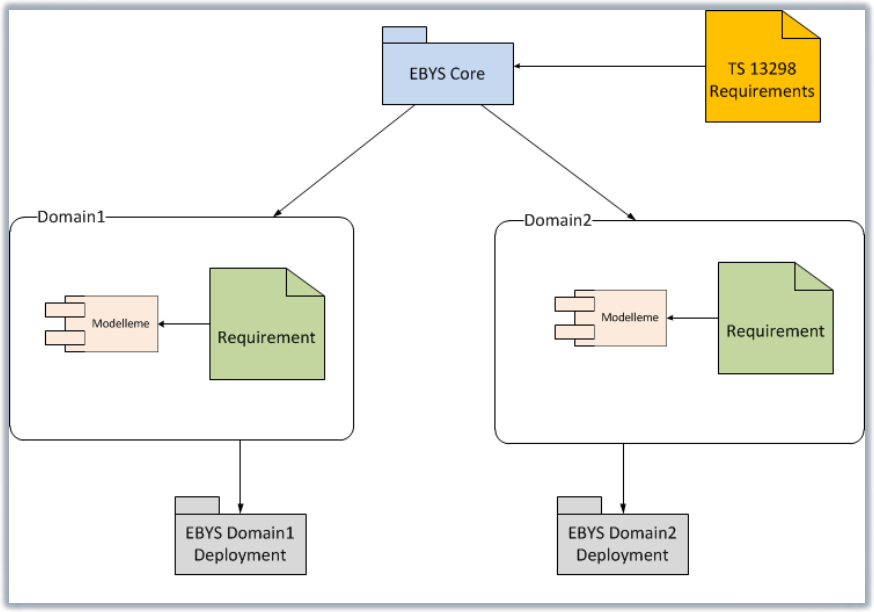
ModelDOC, Şekil 2'de verilen entegrasyon mimarisi temel alınarak tasarlanmıştır. Şekil 1 ile karşılaştırıldığında ModelDOC'un diğer uygulamaların en azından belge yönetimi için gerekli veri kaynaklarına erişerek veri düzeyinde bir entegrasyon sağladığı anlaşılacaktır. Şekil 2'de "Entegrasyon" kutusu içinde diğer uygulamaların verisi modellenerek bir meta veri modeli oluşturulmakta ve bu veri EBYS'nin kendi verisiymiş gibi kullanılmaktadır. Dikkat edilmesi gereken nokta ise diğer uygulamaların bu entegrasyondan haberdar olmasına gerek olmamasıdır ve dolayısıyla veri seviyesi entegrasyonu için herhangi bir ilave servis sunmalarına ve kod güncellemelerine gerek yoktur. Yani, veri seviyesi entegrasyonunda haberdarlık tek yönlüdür. Öte yandan bazı kurumsal uygulamaların verileri ancak sunmuş olduğu servisler aracılığı ile erişilebilir olduğu durumlarda uygulamanın servisi bir veri kaynağı olarak ModelDOC açısından veritabanı yönetim sistemi rolündedir. Böylelikle

veri ve/veya servis paylaşımı olan tüm kurumsal uygulamalar ModelDOC açısından meta veri kaynağıdır. Böylelikle, ModelDOC ile diğer uygulamalar arasında kayıt bazında entegrasyon sağlanmakta ve doküman diğer uygulamalardan alınan bilgileri ve özniteliklerini doküman görünümünde olmasa bile meta veri seviyesinde ilişkilendirerek sorgulamaya izin verebilmektedir.

ModelDOC açısından Şekil 2'de dikkat edilmesi gereken bir husus ise; Stok, Muhasebe vb. uygulamaların kullanıcıları da entegrasyon modeli sayesinde doküman üzerinde işbirliği yapabilecek ve kurumsal verimlilik ve bağ artacaktır. Yani EBYS tek bir birimin değil tüm birimlerden kullanıcıların doğrudan katılımı ile daha etkin kullanılmış olacaktır.

Şekil 2'deki mimari yaklaşım bize EBYS'nin henüz tasarım aşamasında bu entegrasyonu yapması gerektiğini gösterir. Yani, Şekil 1'deki yaklaşımla geliştirilmiş bir EBYS Şekil 2'deki çalışma prensibine geçmesi imkansız değilse bile çok zordur. Çünkü en azından EBYS'nin kaynak kodu üzerinde güncellemeler yapılmalı ve kod tekrar derlenmelidir. Oysa tasarım aşamasında bu husus dikkate alındığında uygulama entegrasyonları meta veri modeli ve SQL sorguları ile yapılabilecek ve EBYS kaynak kodunda herhangi bir güncelleme ve dolayısıyla yeni bir derleme gerekmeyecektir.

Herkes tarafından bilindiği üzere hiçbir kurum diğeri ile aynı değildir ve iş süreçleri farklıdır. Bu yüzden veri entegrasyon gereksinimleri değişmektedir. Çünkü her bir kurumun sahip olduğu kurumsal uygulamalar farklıdır. Bu yüzden aynı EBYS'nin farklı kurumların iş süreçlerine ve veri entegrasyonlarına ayrı ayrı uyarlanması ve kullanıma alınması gerekir. ModelDOC yaklaşımında farklı kurumların entegrasyon ihtiyaçları analiz edilecek ve EBYS yazılımı kaynak kodu güncellenmeksizin (tekrar derlenmeksizin) entegrasyon ihtiyaçları modellenmektedir. Her bir domain (sektör) için oluşturulan bu model ModelDOC'a girdi olarak verildiğinde aynı yazılım derleme gerekmeksizin farklı kurumlarda çalışabilecektir. Anlatılan bu özellik Şekil 3'de kavramsal olarak resmedilmiştir.



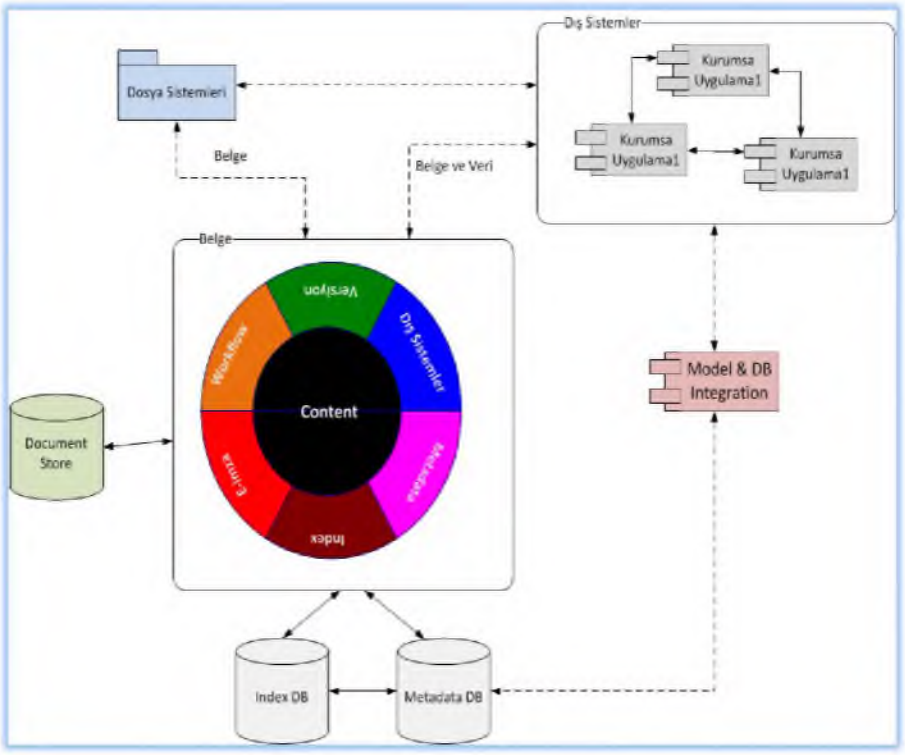
Şekil 3. Farklı kurumlar/sektörler için ModelDOC'un devreye alınması.

EBYS Core olarak adlandırılan generik kısım farklı domain ihtiyaçları modeli ile ayrı ayrı entegrasyonu yerine getirmektedir. Böylelikle yeni bir kurum EBYS alımı için talep ilettiğinde yapılması gereken tek husus, kurumun diğer uygulamaları ile entegrasyon gereklerini modellemektir. Formülize edecek olursak,

ModelDOC Deployment_KurumX = ModelDOC Core + Model_KurumX

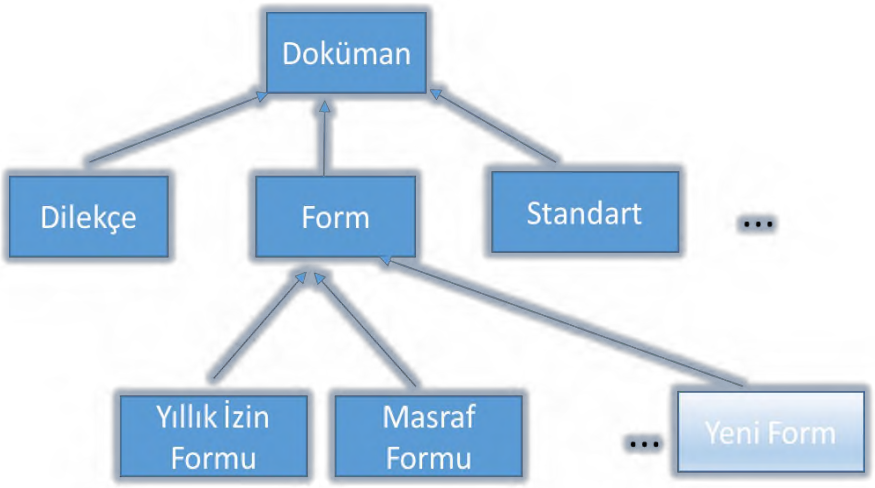
İlave olarak her bir kurum ya da sektöre ait olan modeller kurumsal ihtiyaçlar değiştikçe yine kod değişikliği ve derleme gerekmeksizin çevrimiçi olarak güncellenebilmektedir. Bu şekilde bakım, idame ve yeni sürüme geçme gibi yazılım döngüsü aşamaları hızlı ve zahmetsiz bir şekilde yapılabilmektedir.

Tüm bu entegrasyon vurgusunu karşılamak üzere bir doküman ilişkili olduğu tüm kurumsal uygulamalarla arasındaki bağlantıyı bilmelidir. Şekil 4'de ModelDOC Doküman Modeli gösterilmiştir. Buna göre her bir doküman içeriği, içeriğe kaynaklık eden diğer kurumsal uygulamalarla bağı, saklandığı ortam (CMIS standardında), meta verisi, indeks bilgileri, iş akış bilgileri (BPEL standardı), e-imza bilgileri ve versiyon bilgileri ile beraber bir bütündür.



Şekil 4. ModelDOC Döküman Modeli.

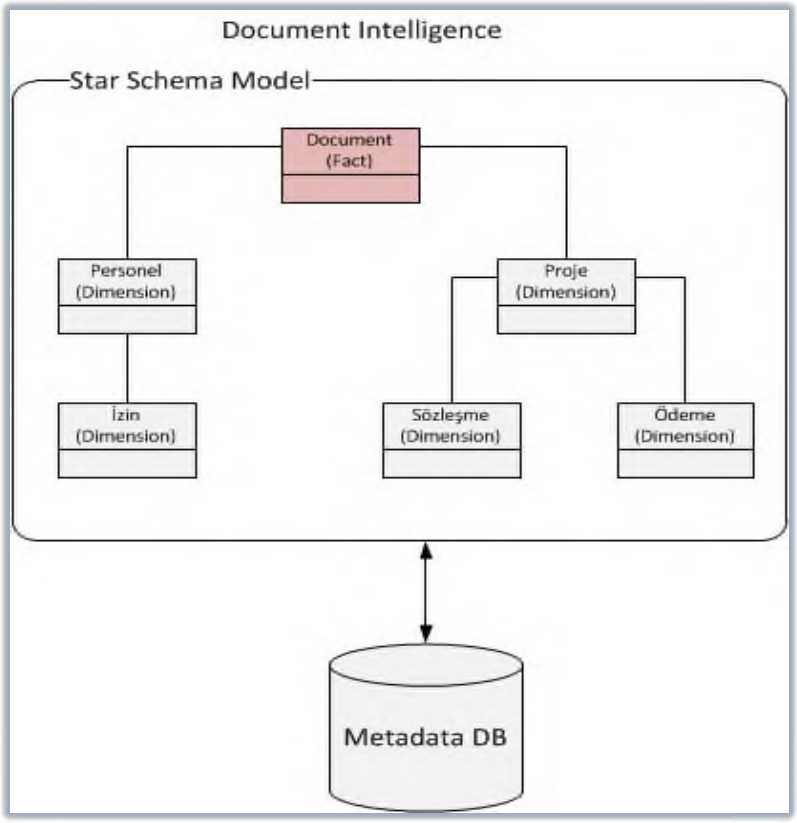
ModelDOC'da doküman sınıfları hiyerarşik olarak modellenmekte ve dokümanın içeriği ve sunumu buna göre yapılmaktadır. Böylelikle yeni doküman sınıfı bu hiyerarşiye istenildiği yerden bağlanarak kodlama gereksinimi olmadan sisteme çevrimiçi olarak tanıtılabilmektedir (Şekil 5).



Şekil 5. ModelDOC Doküman Hiyerarşisi.

Gerek formatlı öznitelikler gerekse serbest metin içerik üzerinden dokümanlar indekslenip sorgulanabilmektedir. Özellikle Solr ve Elasticsearch gibi ölçeklenebilir çözümler bu amaç için uygun araçlardır. ModelDOC gerek ölçeklenebilir doküman yönetimi gerekse hızlı indeksleme ve sorgulama hedefleri nedeniyle bulut tabanlı bir web uygulaması olarak tasarlanmıştır. Bulut tabanlı olması nedeniyle bulutun maliyet ve bakım kolaylığı avantajları (7/24 çalışma, yedekleme) da beraberinde gelmektedir.

EBYS sistemleri temel kullanım olarak kurumun operasyonel ihtiyaçlarına cevap verirler. Oysa sistemden iş zekası amaçlı faydalanmak da mümkündür. Böylelikle kurumun dönemsel performansı EBYS verileri üzerinden takip edilip karar destek amaçlı kullanılabilecektir. ModelDOC dokümana ilişkin meta-veriler (diğer kurumsal uygulama entegrasyon verileri dahil) bir veri ambarında tutulmakta ve OLAP analizleri yapılabilecek yapıdadır. Şekil 6'da basit bir veri ambarının star şeması kesiti örnek olarak verilmiştir.

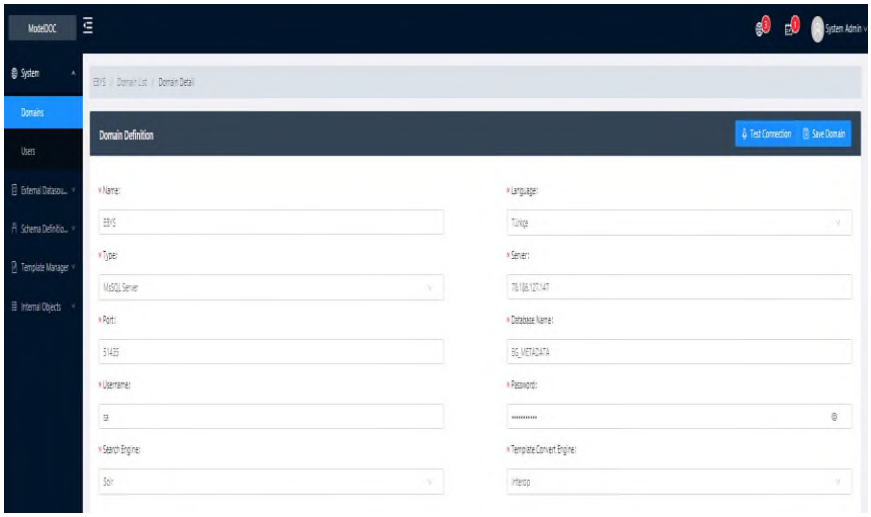


Şekil 6. ModelDOC İş zekası şeması.

4 ModelDOC Geliştirimi

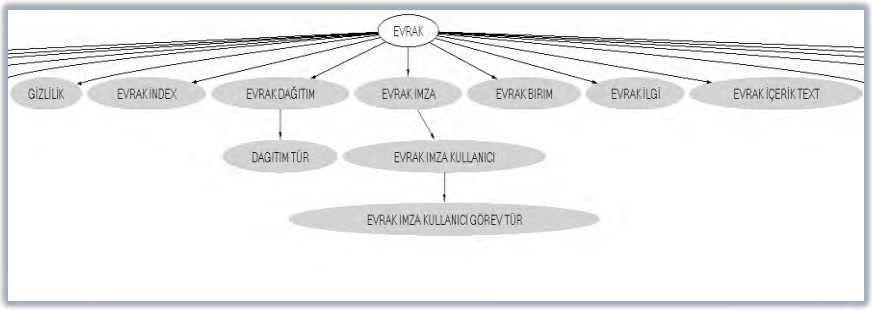
Bölüm 3'de tasarım özellikleri verilen ModelDOC EBYS ürünü Scrum metodolojisi ile geliştirilmiştir. Kaynak kod yönetimi, gereksinim yönetimi, proje yönetimi ve sürüm yönetimi için Microsoft Team Foundation Server kullanılmıştır. Yazılım geliştirme araçları/platformları olarak MS.NET, Angular, Ionic, OpenXML, Hadoop, Apache Solr, Elasticsearch ve Hadoop teknolojileri/ürünleri kullanılmıştır.

Şekil 3'de EBYS Core olarak adlandırılan bileşen temel EBYS işlemlerinin yapıldığı kısımdır. Yine Şekil 3'de gösterilen domainler için modelleme ise ModelDOC Modelleme aracı ile yapılmaktadır. ModelDOC Modelleme aracına domain ve yetkili kullanıcıların tanımlanması ile başlanır. Kullanıcılar rollere atanır. Ardından meta veri modeli oluşturulur ve modelleme tamamlandıktan sonra tüm model (bir ağaç yapısı) indekslenir ve veriler aranabilir hale getirilir. Meta veri modeli şeması XML yapısında saklanır. Şekil 7'de ModelDOC Modelleme aracı arayüzü ve Domain tanım ekranı gösterilmiştir.



Şekil 7. ModelDOC Modelleme aracı arayüzü (Domain tanım ekranı).

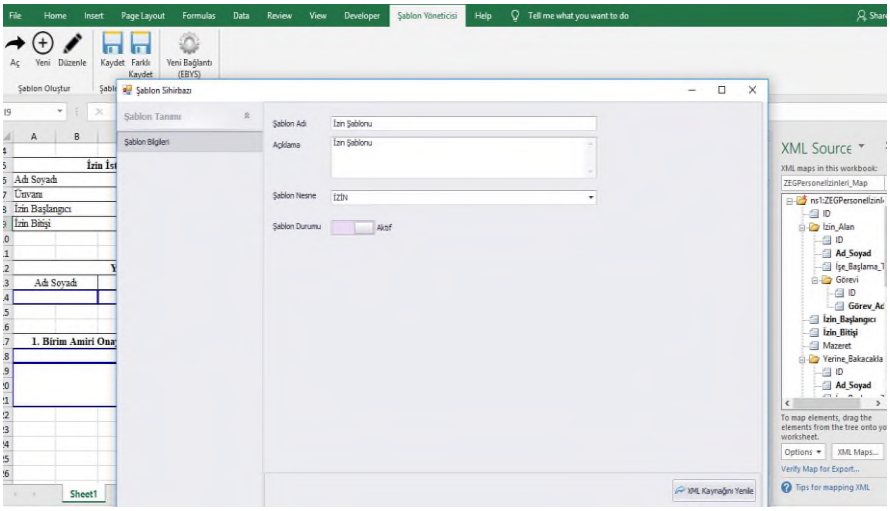
Modelleme için öncelikle yapılması gereken kurumsal uygulama veri kaynaklarının seçilmesidir. Bu noktada çok sayıda Veri Tabanı Yönetim Sistemi'ne ODBC, JDBC ve Native bağlantı desteklenmektedir. Servis tabanlı entegrasyon için ise REST desteklenmektedir. İlave olarak dahili şema yaratılması da desteklenmektedir. Çünkü bazı çok sık ihtiyaç duyulan bilgiler (iller ve ilçeler gibi) hiç bir kurumsal uygulama da olmayabilir. Bu özelliği ile ModelDOC kurumsal veri tamamlama özelliğine de sahiptir. Modelleme kayıt bazında entegrasyon sağlar ve en sonunda ağaç yapısında bir meta veri modeli elde edilir. Bu model ilgili domain için gerekli tüm doküman görünüm, indeksleme, arama ve sorgulama için yeterlidir. Örnek bir meta veri modeli Şekil 8'de verilmiştir. Örnekte evrağın imzacısına (mezuniyeti, doğum tarihi, doğum yeri vb.) ilişkin çok sayıda öznitelik veri modelinde eklenebilir durumdadır.



Şekil 8. ModelDOC entegre meta veri modeli örneği.

Meta veri modelinde dokümanın ilgili olduğu veriler entegre bir şekilde bulunur. Fakat bu bilgilerden hangilerinin doküman görünümünde yer alacağı hangisinin almayacağı, alanlar içinse dokümanın neresinde hangi formatta yer alacağı bilgileri

şablon ve form tasarım aracı ile yapılmaktadır. Şablon ve form tasarım aracı bir Word/Excel Add-in olarak geliştirilmiştir (Şekil 9). Böylelike model-view mantığına göre dokümanın görünümü özelleştirilebilmektedir. Örneğin, imzacının isminde ünvanı yazacak mı yazmayacakmı, sola yanaşık mı sağa yanaşıkmı, yada bunların font büyüklükleri vb. tasarlanmakta ve oluşan görüntü modeli OpenXML olarak saklanmaktadır. Elbette şablon ve form üzerine serbest metin, resim vb. doküman veri modelinde olmayan içerikte eklenebillir. XSLT kullanılarak metin, sayı, tarih, tablo ve liste gibi görünüm ayarları yapılmaktadır.



Şekil 9. ModelDOC Microsoft Excel Şablon Oluşturma Add-in aracı.

Gerek doküman veri modelinde gerekse görünüm veri modelinde yer alan tüm öznitelikler ve serbest metin indekslenmektedir. Böylelikle arama ve sorgulama gerek dokümanın görüntüsü (insan okuyucu) gerekse görüntü haricinde kalan öznitelikleri üzerinden yapılabilmektedir. Her bir domain modeli için indeksleme aracı olarak Apache Solr veya Elasticsearch seçilebilmektedir. Şema çevrimiçi değişebileceği için arama-indexleme motorlarının esneyebilen *dynamic schema* özelliği kullanılmaktadır.

Performans, güvenlik ve güvenilirlik açısından Hadoop dağıtık dosya sistemi tercih edilmiştir. Performans testlerinde 6 milyona yakın doküman indekslenmiş ve arama süresi 1 saniyenin altında gerçekleşmiştir.

Meta veri modeli ağaç yapısında olduğundan veri ambarı tasarımında tercih edilen star şema ile örtüşmektedir. Veri modeli CWM meta-veri modeli standardında OpenXML olarak saklandığı için bu formatı destekleyen herhangi bir İş Zekası aracı tarafından analiz edilebilir durumdadır. Mevcut durumda ZEG Teknoloji BI.NET aracı iş zekası amacıyla ModelDOC ile entegre çalışabilmektedir.

5 Sonuç

ModelDOC tasarımdan gerçekleştirime kadar diğer kurumsal uygulamalarla entegre çalışabilecek ve kurum çalışanlarının katılımcı çalışmasına imkan verecek bir EBYS dir. Böylelikle, EBYS kurum için gerçekten de en önemli IT sistemi olacak ve kurum performansının iyileşmesine ve iş zekası özelliği ile de ölçülmesine imkan sağlayacaktır. Hedef kurum için entegrasyon gereklerinin modellemesi ile elde edilecek model sayesinde kodlama ve derleme gereksinimi olmadan hızlı bir şekilde devreye alınabilecektir. Bu özelliği sayesinde değişen kurum ihtiyaçlarına bile model değişikliği ile kolaylıkla adapte olabilmektedir. *Model-view* ayrımı tasarım yaklaşımı ile doküman şablonunda yer almayan içerikle ilgili öznitelikler de saklandığından zengin indeksleme, arama ve sorgulama imkanı sağlamaktadır.

Teşekkür

ModelDOC projesi TÜBİTAK TEYDEB programı kapsamında desteklenmektedir.

Kaynakça

1. Leikums, T. “A study on electronic document management system integration needs in the public sector”. International Journal of Advances in Engineering & Technology, vol.5, no.1, pp. 194-205, 2012.
2. Haider A. A., Aryati B. and Mahadi B. “Opportunities and Challenges in Implementing Electronic Document Management Systems”. Asian Journal of Applied Sciences, vol.3 issue.1, 2015.
3. TS 13298-Elektronik Belge Yönetimi standardı.
4. Common Warehouse Metamodel, ver 1.1., https://www.omg.org/spec/CWM/About-CWM/, last accessed 2019/06/17.
5. Apache Solr, https://lucene.apache.org/solr/, last accessed 2019/06/17.
6. Elasticsearch, https://www.elastic.co/, last accessed 2019/06/17.
7. Rumbaugh, J., Jacobson, I., Booch, G. “The Unified Modeling Language Reference Manual”. 2nd edn. Addison-Wesley.
8. Sezgin E., Medeni T.D., Bilge M., Önaçan K., Kömürcü R., Dalbay Ö., and Medeni İ.T. “The Perception of Electronic Document Management Systems (EDMS) as a Transformational Information and Communication Technology (ICT) for Public Institutions in Turkey”, In book “Public Administration Reform: Market Demand from Public Organizations”, Edition: 1, Chapter: 17, Publisher: Routledge, Editors: Yogesh K. Dwivedi, Mahmud Shareef, Sanjay K Pandey, Vinod Kumarpp. 279-300, 2013.

Yazılım Mühendisliği Profesyonellerinin Kişisel Öğrenmesi: Türkiye'den bir İnceleme

Selami Bagriyanik[1], Adem Karahoca[2]

[1] Dijital Servisler ve Çözümler Teknoloji Bölümü, Turkcell, İstanbul, Türkiye
selami.bagriyanik@turkcell.com.tr

[2] Yazılım Mühendisliği Bölümü, Bahçeşehir Üniversitesi, İstanbul, Türkiye
adem.karahoca@eng.bau.edu.tr

Özet. Günümüzde, özellikle makine öğrenmesi, yapay zeka ve bulut bilişim gibi teknolojilerdeki çok hızlı gelişmeler üniversitelerin Bilgisayar Bilimleri, Yazılım Mühendisliği, Bilgisayar Mühendisliği ve diğer ilgili bölümlerinden mezun olanların yetkinlikleri ile endüstri beklentileri arasında bir yetenek açığı yaratmaya başlamıştır. Üretilen mesleki bilgilerin kullanım ömrünün her geçen gün kısalması, bu açığı sürekli büyütmeye devam etmektedir. Bahsedilen teknolojilerin tüm sektörler içerisinde kullanımının gitgide artıyor olmasından dolayı söz konusu açık diğer meslek grupları için de geçerli hale gelmiştir. Bu nedenle, yüksek öğrenim sonrası yaşam boyu kişisel öğrenme her zamankinden daha önemli hale gelmiştir. Bu çalışmada, Türkiye'deki büyük bir Teknoloji ve hizmet sağlayıcısı firma ve ekosistemindeki firmalarda çalışan 166 Bilgi ve İletişim Teknolojileri çalışanı ile kişisel öğrenmeleri konusunda gerçekleştirilmiş olan bir anket ve sonuçları ele alınmıştır. Anket sonucunda katılımcıların kişisel öğrenme süreçlerinde hangi mecraları ve içerikleri kullandıkları, yaşadıkları güçlükler ve var olan kısıtları, işyerlerindeki durum ve dramatik teknolojik gelişmelerin sunduğu fırsatlar gibi bilgiler ortaya çıkarılmıştır. Çalışmanın, üniversitelere, öğrencilere, firmalara ve profesyonellere faydalı bilgiler sağlayacağı düşünülmektedir.

Anahtar Kelimeler: Kişisel Öğrenme, Yetenek Açığı, BİT, Anket, Yazılım Mühendisliği, BİT Çalışanları

Personal Learning of Software Engineering Professionals: A Survey From Turkey

Selami Bagriyanik[1], Adem Karahoca[2]

[1] Digital Services and Solutions Technology Department, Turkcell, İstanbul, Turkey
selami.bagriyanik@turkcell.com.tr

[2] Software Engineering Department, Bahcesehir University, İstanbul, Turkey
adem.karahoca@eng.bau.edu.tr

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Abstract. Today, very rapid advances, especially in technologies such as machine learning, artificial intelligence and cloud computing have started to create a skills gap between the competencies of the graduates of Computer Science, Software Engineering, Computer Engineering and other related departments of the universities and the expectations of the industry. The shortening of the lifetime of the produced professional knowledge continues to widen this gap. As the use of the mentioned technologies is increasing in all sectors, this deficit has become valid for the other occupational groups as well. Therefore, lifelong personal learning after higher education has become more important than ever. In this study, a survey has been conducted with 166 Information and Communication Technologies (ICT) professionals working in a large technology and services company and companies in its ecosystem in Turkey. The survey aimed to identify their personal learning perspective. As a result of the questionnaire, information such as which media and content the participants used in their personal learning processes, the difficulties they experienced and their constraints, the situation in the workplace and the opportunities offered by the dramatic technological developments were revealed. It is considered that the study will provide useful information to universities, students, companies and professionals.

Keywords: Personal Learning, Skills Gap, ICT, Survey, Software Engineering, ICT Professionals

1 Giriş

Yapay Zeka alanında yakın geçmişte yaşadığımız dramatik gelişmeler ulaşım, sağlık, bilim, askeri ve finans gibi alanları yeniden şekillendirerek modern hayatı dönüştürmeye başlamıştır [1]. Büyük ölçekli firmalarda çalışanlar veya kendi işlerini büyütmeye çalışan girişimciler artık sadece diğer şirket ve bireylerle rekabet etmemektedir; Nesnelerin Endüstriyel Interneti gibi Yapay Zeka destekli teknolojik yaklaşımlar iş modellerine ve mesleklerine potansiyel tehditler oluşturmaya başlamıştır [2]. Söz konusu gelişmeler henüz otomasyonu görece daha kolay olan meslek gruplarını daha temelden etkiler gibi görünse de bilişim teknolojileri ve etrafındaki disiplinleri de önemli şekilde etkilemeye başlamıştır. Bu gelişmelere koşut olarak mesleki bilgilerin sürekli değişime uğraması veya kullanımdan kalkıp yerini başka bilgilere bırakması döngüsü bilişim alanında özellikle kısalmaya başlamış, üniversitelerin dört senelik mezun verme periyodu endüstri ihtiyaçlarının gerisinde kalmaya başlamış ve bir yetenek açığı oluşmaya başlamıştır. Toffler'in "Geleceğin okuma yazma bilmeyenleri okuyamayanlar değil öğrendiklerini unutmayı ve yeniden öğrenmeyi bilmeyenler olacaktır." [3] sözü bilişim profesyonelleri için her zamankinden daha fazla geçerli hale gelmiştir. Bu nedenle, yüksek öğrenim sonrası yaşam boyu kişisel öğrenme ve yeni bilgilerle sürekli güncellenme profesyoneller için bir zorunluluk halini almıştır. Bu çalışmada, kişisel öğrenme ile ilgili olarak Türkiye'deki durum incelenmiştir. Bu kapsamda, İstanbul merkezli büyük ölçekli bir Teknoloji ve hizmet sağlayıcısı firma ve birlikte çalıştığı tedarikçi ve iş ortağı firmalarda çalışan 166 bilişim profesyoneli ile kişisel öğrenmeye bakış açıları, pratikleri, yaşadıkları güçlükler, gördükleri fırsatlar ve demografik bilgilerine yönelik bir anket gerçekleştirilmiştir. Sözü edilen büyük ölçekli firma, yurt içi ve

yurt dışında faaliyet gösteren, 1000 kişinin üzerinde BİT çalışanı bulunan ve telekomünikasyonun yanısıra finans, sağlık, eğitim gibi bir çok farklı alanda da teknolojik hizmetler ve ürünler geliştiren bir firmadır. Yakın tarihli literatür incelendiğinde Türkiye'de BİT çalışanlarına yönelik bir kişisel öğrenme anket araştırmasına rastlanmamıştır. Çalışmanın bundan sonrasında sırasıyla Anket çalışmasında kullanılan yöntem, anketin araştırma soruları, tasarımı, anketin detaylı sonuçları ve son olarak da araştırmanın sonuçları ve kısıtları verilecektir.

2 Anket

Bu bölümde araştırmada kullanılan yöntem, anketin amacı, tasarımı ve gerçeklenmesine ilişkin bilgiler sunulmuştur.

2.1 Yöntem

Bu anket araştırmasında hedef/soru/metrik paradigması [4] kullanılmıştır. Anketin amacı, Türkiye'deki Bilgi ve İletişim Teknolojisi uygulayıcılarının temel kişisel öğrenme uygulamalarını, zorluklarını ve ihtiyaçlarını anlamaktır. Bu amaç, daha önce yapılmış olan öncül bir mülakat çalışmasının [2] çıktıları kullanılarak türetilmiştir. Anket, sözü edilen çalışmanın mülakatlarından elde edilen içgörülerle tasarlanıp soruların oluşturulmasında da faydalanılmıştır. Sözü edilen çalışmada, 10 bilgi ve iletişim teknolojileri çalışanı ile 45'er dakikalık kişisel öğrenme konulu yarı yapılandırılmış mülakatlar yapılmıştır; sonrasında, kayıt edilen mülakatlar tematik analiz yöntemi [5] ile analiz edilmiştir [2]. Bahsedilen çalışmada elde edilen temalar ve kodlar Tablo 1'de verilmiştir.

Tablo 1. Mülakat İçeriğinin Tematik Analizi Sonrası Tema ve Kodlar.

Temalar	Kodlar	Frekans
Değerlendirme & Ölçüm	Ölçüm, Yetenek Açığı, Gelişim Analizi, Öğreneni Takdir, Öğrenene Geribildirim	3
İşbirliği ve Sosyal Öğrenme	Sosyal Platformlar, Forumlar, Stackoverflow, Topluluk, Katkı, yansıtma, İtibar, Bloglar, Twitter, Öğretme, Katkı, Soru Sorma, Etkileşimli, Başarının Yayılması, Başarısızlık Pratikleri, Facebook, Whatsapp, BİP (Bir mesajlaşma platformu), Skype, Sosyal Çevre	25
Kısıtlar	Zaman Kısıtı, Çevresel Kısıtlar, Kısıtlar, Güvenlik Kısıtları	5
Etik, Gizlilik ve Teori	Etik, Gizlilik Endişeleri, Pedagoji, Androgoji, Sağlık, Teknoloji Dengesi, Bağımlılık	3
Dış Öğrenme Ortamı	Google, Udemy, Coursera, Internet Kaynakları, Amazon, Dijital Öğrenme, Ted, Youtube, Üniversite, Yüksek Öğrenim, Javacodegeeks, Okuyarak Öğrenme, Kindle, W3schools, HBR, Quora, Dice.com, futurism.com, mashable.com, techcrunch.com, Hobi	51
Birlikte Çalışabilirlik	RSS, Entegrasyon	2
Formal Öğrenme	Kurumsal Öğrenme, Zorunlu Öğrenme, Sınıf, Öğretimsel Öğrenme, K12	7
Öğrenme Kaynağı	Video İçeriği, Basılı Kaynak, Bilgi Yönetimi, Hafif Dokümantasyon, Kitap, PDF, Kütüphane, İngilizce Dil Yeterliliği, Problem,	27

	İhtiyaç Tanımı, Powerpoint, Ücretli İçerik, Kodlama Mülakatı, Kitap İncelemesi, Etkileşimli İçerik, Ders İçeriği	
Öğrenme İçeriğinin Güvenilirliği	Kaynak Kredibilitesi, Güvenilirlik, Kalite, Çöp Bilgi, Yazar Kalitesi, uzmanlık, En İyi Uzman, Marka, Kaliteli İçerikler, İçerik Yeterliliği, Güncel İçerik	11
Öğrenme İçeriğinin Aranması	Arama, Sorgu Tasarımı, Kaynaklara Hızlı Erişim, Etiketleme, Uygun Kaynak	18
Mentorluk, Koçluk, Öğretmenlik	Mentor, Koç, Uzman, Öğretmen, Kariyer Planlama, Önyargılı Öğretmen, Uzmanların Bilgi Paylaşımı, Ego	16
Motivasyon	Motivasyon, Oyunlaştırma, Sertifika, Teşvik edici, Ödül, İş-Özel Hayat Dengesi, Yapılacaklar Listesi, Öğrenilecekler Listesi, Basitlik	20
Öğrenme Yolculuğu Tasarımı	Kişiselleştirilmiş Kurumsal Öğrenme, Artımlı Seviye, Aşamalı Öğrenme, Öğrenme Yolculuğu Örüntüleri, Müfredat	5
Pratik & Simülasyon	Pratikle Öğrenme, HackerRank, Problemle Öğrenme, Vaka, Proje, Pluralsite, Pratik, Simülasyon, Sanal, IAAS Ortamlar, Deneysel, Keşfederek Öğrenme	12
Öğrenme Teknolojileri	Öğrenme Asistanı, Sanal Gerçeklik, Yüz yüze, Videokonferans Öğrenmesi, Mobilite, Artırılmış Gerçeklik, Yapay Zeka (YZ), Kişisel ve Kurumsal, Öğretimsel Öğrenme Dengesi, Büyük Veri, Gerçek Zamanlı Öğrenme, Dikey ve Yatay Öğrenme, uzmanlaşma, İnsan vs YZ'ye karşı, Anlık Öürenme, Adaptif, Bağlamsal	37
İşe alım Platformları	LinkedIn, Geçim	3
İşyerinde Öğrenme	İşyerinde Öğrenme, Çalışırken, Gömülü Öürenme, Mavi yaka öürenmesi, Formal Öğrenme, Formal Olmayan Öğrenme, Yüksek Öğrenim, Destek, İşyeri Öğrenmesine Paralel Faydalar, Ağ Öğrenmesi	13

2.2 Araştırma Soruları

Bir önceki alt bölümde oluşturulan araştırma amacını baz alarak aşağıdaki altı araştırma sorusu (AS) oluşturulmuştur:

AS 1: Anket katılımcılarının profilleri ve demografileri nelerdir?
AS 2: Kişisel öğrenme için ne tür yerler, kanallar ve içerik kullanılmaktadır?
AS 3: Kişisel öğrenme sürecinde ne tür zorluklar ve kısıtlamalar yaşanmaktadır?
AS 4: Örgün öğrenmenin şimdi algılanan önemi nedir ve gelecekte ne olacaktır?
AS 5: İşyerinde kişisel öğrenim desteklenmekte midir?
AS 6: Hangi tür teknolojiler kişisel öğrenme için fırsatlar sunmaktadır?

2.3 Anket Tasarımı

Bu çalışmada, anketlere geliştiricilerin katılım oranlarının arttırılmasıyla ilgili bir çalışmadaki görüş ve önerilerden [6] faydalanılmıştır. Bu çalışmadaki durumda pratikte mümkün olmadığı için olasılıklı bir örnekleme yöntemi [7] kullanılmamıştır. Anket soruları, alt bölüm 3.1'deki tematik analizden oluşturulan temalar ve kodlar dikkate alınarak türetilen araştırma sorularından elde edilmiştir. Anket soruları türleri ile birlikte Tablo 2'de gösterilmektedir. Anket, Survey Monkey (www.surveymonkey.com) adlı çevrimiçi bir anket sitesinde yayınlanmıştır. Katılım oranını artırmak için, şirketin ekosistemindeki ilgili bölümlere ve diğer iş ortağı ve tedarikçi firmalara e-posta davetiyesi gönderilmiş ve linkedin gibi sosyal mecralardan duyurular

yapılmıştır. Katılım oranını arttırmak için ankete katılanlara anket henüz yayınlanmadan sonuçlarının kabaca paylaşılacağı bildirilmiştir. Bu amaçla isteyen katılımcılar e-posta adreslerini anket içerisinden paylaşabilmişlerdir. Anket, 2017 ortalarında iki ay boyunca katılımcılara açılmıştır. Ankete katılan bilgi ve iletişim teknolojileri uzmanlarının toplam sayısı 166 olarak gerçekleşmiştir.

Tablo 2. Anket Soruları.

AS	Soru No & Soru	Tek Yanıt	Çoklu Yanıt	Likert Skalası	Sayı	Serbest Metin
1	S1. Hangi seçenekler çalışma alanınızı en iyi şekilde tanımlar?		x			x
1	S2. Hangi seçenekler rolünüzü en iyi şekilde tanımlar?		x			x
1	S3. Yazılım endüstrisinde kaç senelik tecrübeniz bulunmaktadır?				x	
1	S4. Aldığınız en yüksek Akademik derece nedir?	x				
1	S5. Hangi Lisans dereceleriniz bulunmaktadır?		x			x
1	S6. Hangi şehirlerde yaşıyorsunuz?		x			x
1	S7. Şirketiniz hangi sektörde faaliyet göstermektedir?		x			x
1	S8. Şirketinizde kaç kişi çalışmaktadır?	x				
1	S9. Cinsiyetiniz nedir?	x				
2	S10. Mesleki gelişiminiz için hangi siteleri ziyaret edersiniz?		x			x
2	S11. Mesleki gelişiminiz için hangi tipteki öğrenme içeriklerini kullanırsınız?		x			x
2	S12. Sosyal ve işbirliğine dayalı (Fikir alışverişi veya öğrenilenlerin yansıtıldığı ortamlar, blog'lar, vb.) öğrenmeyi kullanır mısınız?			x		
3	S13. Kişisel öğrenme sürecinizde yaşadığınız en önemli güçlükler nelerdir?		x			x
2	S14. Kişisel öğrenmenizi en çok nerelerde gerçekleştirirsiniz?		x			x
4	S15. Sizce Örgün Eğitim önemli midir?			x		
5	S16. Firmanız tarafından sağlanan öğrenme imkanları kişisel öğreniminizi ne derece desteklemektedir?			x		
3	S17. Kişisel öğrenme sürecinizde bir mentor/koç/danışman/öğretmen sahibi olmak sizce önemli midir?			x		
6	S18. "Yeni gelişen teknolojilerle ve E-Öğrenme ile desteklenen kişisel öğrenme			x		

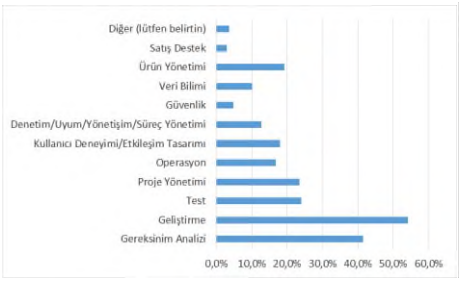
	gelecekte geleneksel/örgün eğitimin yerini alacaktır.” ifadesine ne ölçüde katılırsınız?			
6	S19. Hangi yeni gelişen teknolojiler kişisel öğrenme için fırsatlar sunmaktadır?	x		x
6	S20. “Büyük veri ve Yapay Zeka kişisel öğrenme için en büyük potansiyele sahiptir.” ifadesine ne ölçüde katılırsınız?		x	
	S21. Kişisel öğrenme sürecinizle ilgili olarak eklemek istediğiniz yorumlarınız var mı? Açıklayabilir misiniz?			x
	S22. Araştırma sonuçlarını sizinle paylaşmamızı isterseniz e-mail adresinizi paylaşabilir misiniz?			x

3 Anket Sonuçları

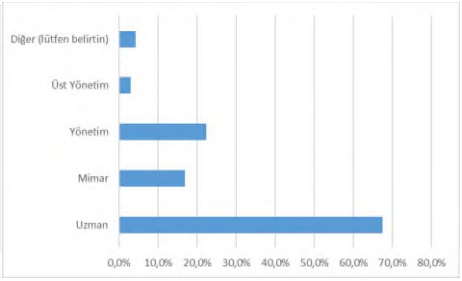
Bu bölümde anket sonuçları grafiklerle verilmiş ve gözlemler paylaşılarak yorumlanmıştır.

3.1 Katılımcıların Çalışma Alanları (S1)

Bir yazılım pratisyeni şirketinde çeşitli sorumluluklara sahip olabileceğinden, bu soru çoklu cevap sorusu olarak tasarlanmıştır. Şekil 1’den görüleceği gibi Yazılım Geliştirme ve İhtiyaç Analizi katılımcıların en fazla bulundukları alanlardır.



Şekil 1. Çalışma Alanları



Şekil 2. Roller

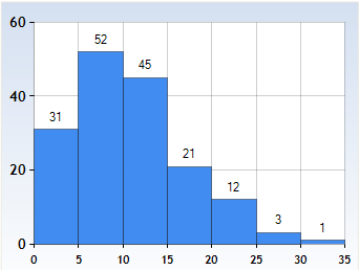
3.2 Katılımcıların Rolleri (S2)

Bu da çok cevaplı bir sorudur. Roller Şekil 2’de gösterilmiştir. Ankete katılanların çoğu Uzman rolünde bulunmaktadır. Bir katılımcının birden fazla rol seçtiği vakaların sıklığını da ölçtük. Ankete katılanların % 11'i aynı anda hem Uzman hem de Mimar olarak çalışmaktadır. İlginç bir şekilde, anket katılımcılarının % 2'si Uzman, Mimar ve Yönetim rollerinin tümünü aynı anda gerçekleştirmektedir.

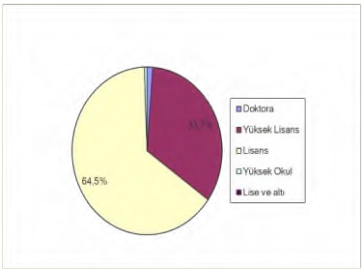
3.3 Katılımcıların Deneyimleri (S3)

Katılımcıların iş tecrübelerinin dağılımı (yatay eksende yıl olarak) Şekil 3’te histogram olarak gösterilmektedir. Ortanca ve ortalama değerler sırasıyla 9 ve 10'dur. Bu, ankete

katılan genç ve tecrübeli BİT uzmanlarının büyük ölçüde dengeli bir dağılımının olduğunu göstermektedir.



Şekil 3. Deneyimler



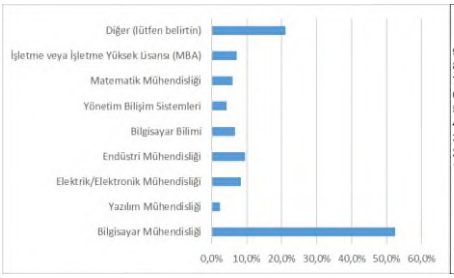
Şekil 4. Akademik Dereceler

3.4 Katılımcıların Akademik Dereceleri (S4)

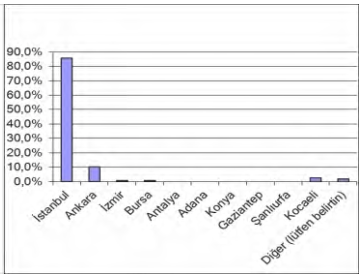
Şekil 4'ten görüldüğü gibi katılımcının en yüksek akademik derecelerinin çoğunluğu Lisans ve Yüksek Lisans'tır. Katılımcılar arasında lise veya altı okul mezunu yoktur.

3.5 Katılımcıların Lisans Dereceleri (S5)

Lisans derecelerinin dağılımı Şekil 5'ten incelenebilir. Bilgisayar Mühendisliği, beklendiği gibi, katılımcıların Lisans derecelerinde en yüksek orana sahiptir. Yazılım Mühendisliği az bir oran ile göze çarpmaktadır. Kayda değer bir bulgu daha gözlenmektedir. "Diğer" kategorisinin % 20 gibi görece yüksek bir değerde olması ilginçtir. Diğer kategorisinde Makine Mühendisliği gibi diğer mühendislik disiplinleri ve Hukuk gibi sosyal bilimler alanları da bulunmaktadır. Bu sonuç, farklı mesleklerde eğitilmiş birçok kişinin yazılım endüstrisinde çalıştığını göstermektedir.



Şekil 5. Lisans Dereceleri



Şekil 6. Lokasyonlar

3.6 Katılımcıların Bulundukları Yerler (S6)

Katılımcıların çoğunluğu İstanbul ve Ankara'da yaşamaktadır. Bu durumun iki nedeni bulunmaktadır. İlk olarak, BİT şirketleri ağırlıklı olarak bu iki şehirde bulunmaktadır. İkincisi, yazarların yeri İstanbul'dur ve ağlarındaki kişiler de çoğunlukla İstanbul'dadır. Diğer bazı yerlerden minimal katılımlar Şekil 6'dan görülebilir.

3.7 Şirket Sektörü (S7)

Bilgi ve İletişim Teknolojileri, katılımcıların çalıştığı şirketlerin en popüler sektörüdür. Bu beklenen bir sonuçtur zira anket bir Teknoloji ve hizmet firmasında ve onun ekosisteminde gerçekleştirilmiştir.

3.8 Şirket Büyüklüğü (S8)

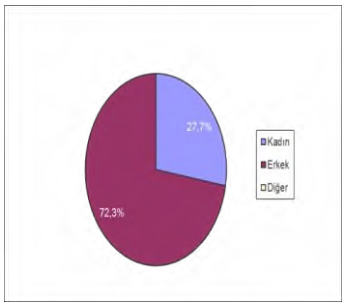
Anket katılımcılarının çalıştığı şirketler genellikle orta (% 51,8) ve büyük (% 32,5) ölçekli şirketler arasındadır.

3.9 Katılımcıların Cinsiyetleri (S9)

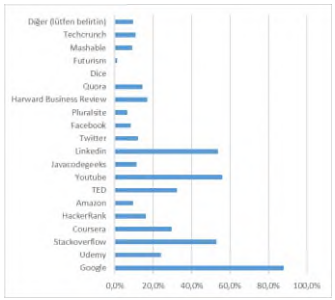
Anket katılımcıları çoğunlukla (% 72,3'ü) erkektir (Şekil 7). Çalışma alanları değiştiğinde dağılımların da önemli ölçüde değişebildiği de gözlenmiştir. Örneğin, yazılım geliştirici katılımcılar arasında erkeklerin oranı % 85 seviyesine çıkmaktadır.

3.10 Ziyaret Edilen Web Siteleri (S10)

Ziyaret edilen sitelerin dağılımlarına Şekil 9'dan bakıldığında Google, öğrenmek için



Şekil 7. Cinsiyetler



Şekil 8. Web Siteleri

kullanılan en popüler sitedir. Katılımcıların yaklaşık % 90'ı kullanmaktadır. Bu durum, bir arama aracının kişisel öğrenmede bulunması gereken olmazsa olmaz bir şey olduğu anlamına gelir. Burada altı çizilmesi gereken bir konu Google'ın doğrudan öğrenme içeriği olan bir web sitesi olarak kullanımından ziyade öğrenme içeriğine erişim için kullanılan bir arama imkanı olarak kullanılmasıdır. Stackoverflow, YouTube ve LinkedIn yazılım uzmanları tarafından % 50'den fazla ziyaret edilmektedir. Geliştiriciler için Stackoverflow oranı % 81'e ulaşmaktadır. Diğer seçeneği için belirtilen bazı siteler de Google Scholar, ResearchGate, Udacity ve Linda gibi platformlar olmuştur.

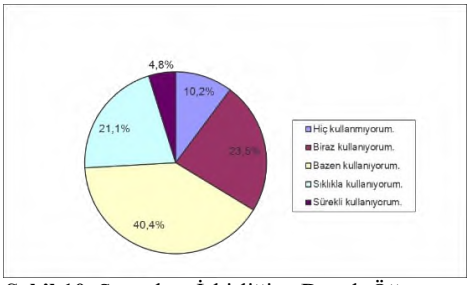
3.11 Öğrenme İçeriği Türleri (S11)

PDF'ler gibi E-Belgeler, katılımcılar tarafından kullanılan en popüler içeriktir (Şekil 9). Bu sonuç şaşırtıcıdır çünkü videolar MOOC'lerde, LMS'lerde ve diğer öğrenim

sitelerinde en yaygın kullanılan içeriktir. Ayrıca, bazı deneysel öğrenme içeriğinin katılımcıların % 40'ından fazlası tarafından kullanılması da dikkat çekicidir.



Şekil 9. İçerik Türleri



Şekil 10. Sosyal ve İşbirliğine Dayalı Öğrenme

3.12 Sosyal ve İşbirliğine Dayalı Öğrenme (S12)

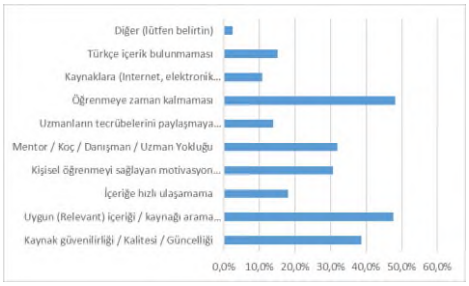
Şekil 10'deki pasta grafik grafiklerinden görülebileceği gibi, katılımcıların önemli bir kısmı (~% 66) sosyal ve işbirlikçi öğrenme toplulukları kullanmaktadır.

3.13 Karşılaşılan Zorluklar (S13)

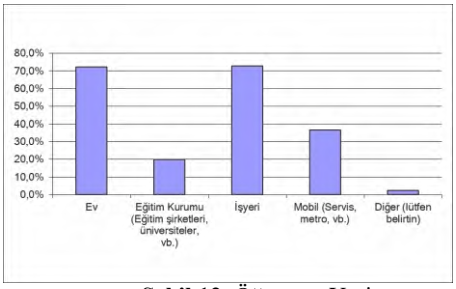
Katılımcıların kişisel öğrenme süreçlerinde karşılaştıkları en önemli iki zorluk, uygun içerik arama süreci (% 47,6) ve öğrenme için zaman eksikliğidir (% 48,2). Şekil 11'ten diğer zorluklar görülebilir. Üçüncü önemli unsur, bulunan içeriğin kalitesi ve güvenilirliğidir. Bu sonuçlar, S10'daki arama motoru gereksinimiyle oldukça uyumlu görünmektedir. Katılımcılar, güncel, kaliteli ve makul derecede alakalı içeriğe hızla erişmek istemektedir.

3.14 Öğrenme Yeri (S14)

Kişisel öğrenme hala ofislerde, evlerde ve eğitim kurumlarında gerçekleşmektedir. Ek olarak, mobil durumlarda önemli miktarda öğrenme gözlenmektedir. Öğrenme yerleri Şekil 12'te gösterilmektedir.



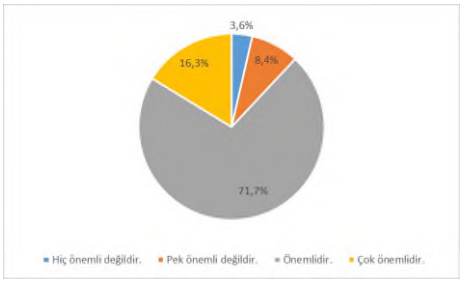
Şekil 11. Zorluklar



Şekil 12. Öğrenme Yeri

3.15 Örgün Eğitim (S15)

Şekil 13'te görüldüğü gibi, Formel Öğrenme, katılımcıların neredeyse tümü için önemli olmaya devam etmektedir. Anketteki diğer cevaplar göz önüne alındığında, bu kişisel öğrenmeye duyulan ihtiyaçla çelişmediğini, aksine geleneksel öğrenmenin kişisel öğrenmeyi tamamlaması gerektiğini göstermektedir. Örneğin, öğrencinin kişisel öğrenme ortamında biriktirdiği bilgiler, endüstri tarafından kabul gören sertifika programları ile sertifikalandırılabilir.



Şekil 13. Örgün Eğitim



Şekil 14. Şirket Desteği

3.16 Şirketin Çalışan Öğrenmesine Desteği (S16)

Firmalar genellikle çalışanlarının kişisel öğrenimlerine yatırım yapar gibi görünmektedir. Şekil 14'ten görülebilecek bu durum, çalışanların kişisel öğrenme süreçlerini ve şirketteki iş başında öğrenmeyi bir araya getirerek yaşam boyu öğrenme için bir fırsat yaratmaktadır.

3.17 Mentor Sahibi Olma (S17)

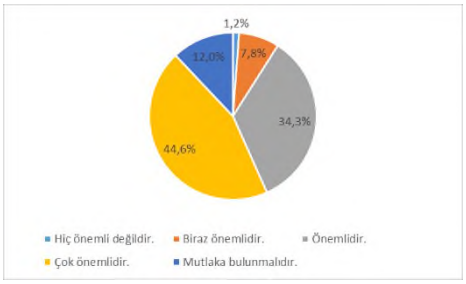
Şekil 15'den görüldüğü gibi hemen hemen tüm katılımcılar, S15'teki Örgün Öğrenim'deki duruma benzer şekilde bir danışman, koç veya öğretmen bulunmasının gerekli olduğu konusunda hemfikirdirler. Bu ihtiyaç daha önce 13. Soruda iki zorluk olarak ortaya çıkmıştı: mentor eksikliği ve motivasyon eksikliği. Bu unsur geleneksel öğrenme ve kişisel öğrenme yaklaşımları için bir başka sinerji alanı olarak dikkat çekmektedir.

3.18 Örgün Eğitimin Geleceği (S18)

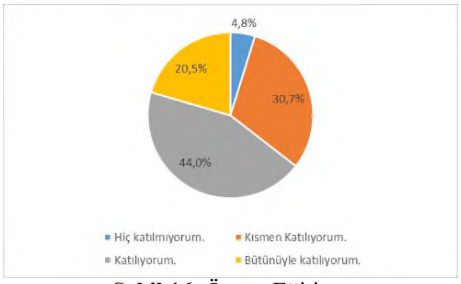
Katılımcıların büyük bir kısmı, gelişen teknolojilerle donatılmış Kişisel Öğrenmenin gelecekte geleneksel öğrenmeyi büyük ölçüde değiştireceğine inanmaktadır. Şekil 16'de gösterildiği üzere katılımcıların sadece ~% 5'i bu görüşe katılmamaktadır.

3.19 Gelişmekte Olan Teknolojiler (S19)

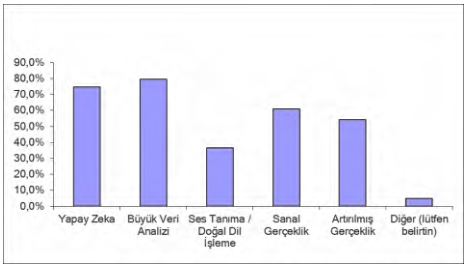
Katılımcılara göre, Yapay Zeka ve Büyük Veri Analizi kişisel öğrenme için en güçlü



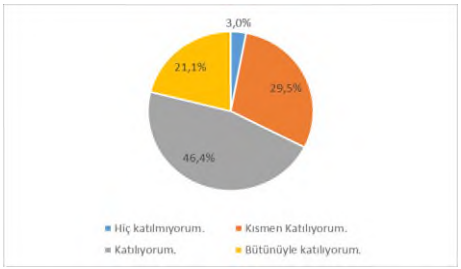
Şekil 15. Mentor Sahipliği



Şekil 16. Örgün Eğitim



Şekil 17. Teknolojiler



Şekil 18. Büyük Veri ve Yapay Zeka

teknolojiler gibi görünmektedir. Şekil 17'dan görüldüğü üzere Sanal ve Artırılmış Gerçeklik ve Konuşmalı Arayüz teknolojilerinin de önemli bir etkisi olacağını düşünmektedirler.

3.20 Büyük Veri Analizi ve Yapay Zeka (S20) ve İlave Yorumlar (S21)

Bu soruda özellikle Yapay Zeka ve Büyük Veri Analizinin Kişisel Öğrenme üzerindeki etkisi katılımcılara sorulmuştur. Katılımcılar çoğunlukla büyük bir etki beklemektedirler (Şekil 18). Katılımcılardan sadece birkaç ekstra yorum yapılmıştır. Yorum sayısı çok sınırlı olduğundan ve yorumlar genel olarak önceki anket yanıtlarının tekrarı olduğu için ek bir tematik analiz yapılmamıştır.

4 Sonuçlar ve Kısıtlar

Bu çalışmada, Türkiye'deki bilgi ve iletişim teknolojileri profesyonellerinin kişisel öğrenmeye dair bakış açıları, pratikleri, yaşadıkları güçlükler ve demografik bilgilerinin anlaşılması amaçlanmış ve bu amaçla 21 soruluk bir anket tasarlanarak uygulanmıştır. Anket amacı ve soruları tasarlanırken daha önce aynı firmada yapılmış olan bir mülakat saha çalışması [2] baz alınmıştır. Ankette öne çıkan konular şöyle özetlenebilir: 1) Bilişim profesyonellerinin büyük bir kısmı erkeklerden oluşmakta, çoğunluğu bilgisayar mühendisliği mezunu olsa da çok farklı bölümlerden de bir çok mezun sektörde çalışmaktadır. 2) Çalışanların kişisel öğrenme farkındalıkları yüksektir ve çeşitli kaynaklardan ve mecralardan edindikleri içeriklerle gelişim süreçlerini sürdürmektedirler. Bu gözlem, yakın zamanda yapılmış olan bir çalışmadaki [8] sadece

teknik yeteneklerin yeterli olmadığı, bunun yanısıra teknik olmayan davranışsal yeteneklerin de (soft skills) çalışma hayatında çok önemli olduğu ve bu yeteneklerden birinin de kişisel öğrenme olduğu sonucu ile örtüşmektedir. 3) Yeni teknolojilerin sunduğu fırsatlarla eğitimin önemli ölçüde dönüşüme uğramakta olduğunu düşünmektedirler. Benzer sonuç, yetenek açığını irdeleyen bazı çalışmalarda da [9] sürekli öğrenme ve güncellenme şeklinde raporlanmıştır. Ek olarak geleneksel yöntemlerin mentorluk gibi güçlü yanlarıyla bir sinerjinin gerekliliğini belirtmektedirler.

Çalışmada ortaya çıkan sonuçlar değerlendirildiğinde üniversitelerin, eğitim programlarını ve eğitsel yaklaşımlarını güncel gelişmelere ve endüstri ihtiyaçlarına göre daha dinamik düzenlemeleri, öğrencilerin kişisel öğrenme konusundaki farkındalıklarını eğitim hayatları devam ederken arttırmaları ve kişisel öğrenme patikalarını oluşturmaları, firmaların kurumsal merkezli geleneksel sınıf eğitimlerinin yanısıra kişisel öğrenmeyi destekleyici adımlar atmaları (Örneğin, Udemy veya Udacity gibi platformlardaki bazı güncel içeriklere ücretsiz erişimlerini sağlamak ve eğitimleri başarılı bir şekilde tamamlayanları ödüllendirmek, bir sonraki kariyer hareketleri ile ilişkilendirmek gibi.) ve bu yaklaşımı benimseyen çalışanları için uygun ortamı sağlamaları önerilebilir. Anket tasarımı ve uygulanması, özellikle İstanbul'da bulunan büyük bir Teknoloji firmasının ekosistemindeki katılımcılarla yürütülen saha çalışmaları kullanılarak yapılmıştır. Bu nedenle, çalışmanın Türkiye'deki BİT profesyonellerinin tamamının görüşlerini temsil etmemesi kısıt olarak belirtilebilir.

Referanslar

[1] K. Grace, J. Salvatier, A. Dafoe, B. Zhang, O. Evans, When Will AI Exceed Human Performance ? Evidence from AI Experts, Arxiv.Org. (2017) 1–21.

[2] S. Bagriyanik, A. Karahoca, D. Karahoca, A New Proposal Architecture for Reference Ple: Refple, Int. J. Mech. Eng. Technol. 10 (2019) 1814–1822.

[3] Wikipedia, Alvin Toffler, (2017). https://en.wikipedia.org/wiki/Alvin_Toffler.

[4] V.R. Basili, Software modeling and measurement: the Goal/Question/Metric paradigm, 1992.

[5] V. Braun, V. Clarke, Using thematic analysis in psychology, Qual. Res. Psychol. 3 (2006) 77–101.

[6] E. Smith, R. Loftin, E. Murphy-hill, C. Bird, Improving developer participation rates in surveys, in: Coop. Hum. Asp. Softw. Eng. (CHASE), 2013 6th Int. Work., IEEE, 2013: pp. 89–92.

[7] B. Kitchenham, S.L. Pfleeger, Principles of survey research part 5 : populations and samples, ACM SIGSOFT Softw. Eng. Notes. 27 (2002) 17–20.

[8] V. Garousi, G. Giray, E. Tuzun, C. Catal, M. Felderer, Closing the gap between software engineering education and industrial needs, IEEE Softw. (2019).

[9] J. Liebenberg, M. Huisman, E. Mentz, Software: university courses versus workplace practice, Ind. High. Educ. 29 (2015) 221–235.

Yazılım Uzmanları için Tasarlanmış Dijital Olmayan İş Birliği Oyunu

Hazal İpek Kahraman[1], Yasemin Bektürk[1], Irmak Angün[1], Yasemin Yiğit Kuru[1] ve Murat Yılmaz[2]

[1] Huawei Ar-Ge Merkezi, İstanbul, Türkiye
{hazal.ipek.kahraman,yasemin.bekturk,irmak.angun,yasemin.yigit.kuru}@huawei.com
[2] Dublin Şehir Üniversitesi, Bilgisayar Bilimleri ve Mühendisliği Bölümü, Dublin, İrlanda
murat.yilmaz@dcu.ie

Özet. İnsanlık tarihi kadar eski olan dijital olmayan oyunlar bireylerin hayatında özellikle sosyal etkileşimi güçlendirmek adına önemli bir yer tutmaktadır. Kişiler bu oyunlar yardımıyla daha etkin bir biçimde etkileşim kurmakta ve kendilerine verilen görevleri daha hızlı bir biçimde kavrayabilmektedirler. Çevik yazılım geliştirme süreçleri diğer mühendislik disiplinlerinde tanımlanan üretim süreçlerine göre daha fazla sosyal etkileşim içermektedir. Çevik dönüşüm konusunda bir yıldan fazla süredir çalışmalarını sürdüren Huawei Türkiye AR-GE Merkezi bu dönüşümü sağlamak adına yürüttüğü eğitim ve çalıştaylarda dijital olmayan oyunları aktif olarak kullanmaktadır. Bu oyunlar sayesinde yazılım takımlarındaki çevik dönüşüm faaliyetlerinin önemli bir ivme kazandığı gözlemlenmektedir. Huawei Türkiye AR-GE Merkezi bünyesindeki farklı departmanlarda çalışmakta olan kişilerin iş birliği yeteneklerinin artırılması, bilgi paylaşım kabiliyetlerinin iyileştirilmesi ve dolayısıyla yazılım takımlarının sosyal etkileşim becerilerinin artırılması hedeflenmektedir. Bu deneyim bildirisinde belirlenen hedefler doğrultusunda Huawei Türkiye AR-GE Merkezi Kalite ve Operasyon Departmanı tarafından yazılım uzmanlarına özel olarak tasarlanmış (dijital olmayan) bir oyun tanıtılacak ve endüstriyel ortamda edinilen deneyimler paylaşılacaktır.

Anahtar Kelimeler: Yazılım Bilgi Oyunu, İnsan-insan etkileşimi, Dijital Olmayan Oyunlar, Çevik Yazılım

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

A Non-Digital Collaboration Game Designed for Software Professionals

Hazal Ipek Kahraman[1], Yasemin Bekturk[1], Irmak Angun[1], Yasemin Yigit Kuru[1] and Murat Yilmaz[2]

[1] Huawei R&D Center, Istanbul, Turkey
{hazal.ipek.kahraman,yasemin.bekturk,irmak.angun,yasemin.yigit.kuru}@huawei.com
[2] Dublin City University, School of Computing, Dublin, Ireland
murat.yilmaz@dcu.ie

Abstract. Non-digital games, which are as old as human history, play an essential role in strengthening social interaction, especially in the life of individuals. People interact more effectively with the help of these games and understand the tasks assigned to them more quickly. Agile software development processes involve more social interaction than production processes described in other engineering disciplines. Huawei Turkey R&D Center has been working on Agile transformation during last year by conducting various training and workshop sessions by using non-digital games. By the help of these games, it is observed that the agile transformation activities in the software teams gained a significant acceleration. Huawei Turkey R&D Center of the people working in different departments within the cooperation in improving the skills, improving information sharing capabilities of social interaction and thus aims to increase the demand for software teams. In line with the objectives set out in this paper, we introduce a non-digital game for software professionals, which was developed by Huawei Turkey R&D Center Quality and Operations Department. This paper will also share the results and experience gained by the realization of the game on an industrial setting.

Keywords: Software Knowledge Game, Human-Human Interaction, Non-Digital Games, Agile Development

1 Giriş

Oyunlar insanlık tarihi kadar eski kurgulardır. Bu kurgular sosyal yaşamın ayrılmaz bir parçası olarak görülmekte olup, bireylerin sosyal ve örgütsel etkileşimini artırma, kendini ifade etmesini kolaylaştırma ve bireylerin eğitim almalarını kolaylaştırma amacıyla kullanılmıştır [1]. Bu amaç doğrultusunda eğlence, eğitim, iletişim vb. gibi birçok farklı durum için iki veya daha fazla birey arasında gerçekleştirilen stratejik sosyal etkileşim ve planlanan bir hedef için gerçekleştirilen aktivitelere oyun adı verilir [2]. Oyunlar bireylerin stratejik karar verme

yeteneğini geliştiren ve kendilerini ispat edebilecekleri bir mecra olarak da karşımıza çıkmaktadır. Oyun kurgusal olduğu bilinen ve gündelik hayatın dışında yer alan, bununla birlikte oyuncuyu da tamamen içine çeken, gönüllü, özgür bir eylemdir [3]. Sınırları özellikle belirlenmiş zaman ve mekân içinde gerçekleşen, genellikle maddi çıkarlardan uzak bu eylem, tanımlanmış kurallara göre, belli bir düzen içinde gerçekleştirilir [4]. Oyuncular; katılımcılar, bireyler veya takımlar olarak belli kurallar içinde etkileşip kişiliklerine uygun bir strateji profili oluştururlar [5]. Farklı birçok şekilde tanımlanabilen oyun kavramı, tarih boyunca hayatın her alanında kültürün temel taşlarından birisi olarak varlığını sürdürmüştü [6]. Oyun kavramı ve bu kavramın uygulamaları psikoloji [7], bilgisayar bilimleri [8] ve sosyal bilimler [9] gibi birçok alanda karşımıza çıkmaktadır. Oyunlar, motivasyon artırıcı bir etkiye sahip olmasının yanı sıra verimliliğin daha fazla olduğu, katılımcıların etkileşimlerinde artışın gözlendiği, etkileşimlerin artmasının sonucu olarak bilgi aktarımının çoğaldığı ve katılımcıların zamanını daha etkin bir şekilde kullanabildiği bir ortam sağlar [10, 11]. Dijital olmayan oyunlar, teknolojinin insanlar arasındaki ilişkileri zaman zaman körelttiği yazılım sektöründe; sosyal ortamlar kurarak katılımcıların birbirileri ile sağlıklı iletişim kurmalarına, beyin jimnastiği yapmalarına, somut materyaller ile görsel hafızanın kuvvetlendirilmesine ve kişisel çıkarları düşünmeksizin ortak çıkarlar doğrultusunda işbirliğini destekleyen birleşimler oluşmasına olanak sağlar. Dijital olmayan oyunlar; popülerleşen oyunların aksine ihtiyaca yönelik olarak, sorun giderme, iyileştirme vb. gibi amaçlarla belirli bir kitleye yönelik olarak tasarlanabilmektedir [4, 12]. Yazılım mühendisliğindeki dijital olmayan oyunlar ile ilgili örnekler incelendiğinde [13], bu çalışmalardan elde edilen çıkarımların yazılım geliştiriciler tarafından tanımlanmış, potansiyel pek çok problemin çözümünde kullanılabildiği gözlemlenmiştir [14, 15]. Huawei Türkiye AR-GE Merkezi Kalite ve Operasyon Departmanı proje takımlarına sağladığı destekler esnasında, farklı proje takımlarında yer alan kişiler arasındaki iletişim ve paylaşım eksikliği gibi problemlerin farkına varmıştır. Bu problemlerin giderilmesi için çeşitli araştırmalar yapılmıştır. Araştırmalar sonucunda elde edilen geri dönüşler ile bu eksikliklerin kişiler arasındaki bilgi paylaşımının, çalışanların verimliliğinin, sosyal etkileşimlerinin ve motivasyonlarının yanı sıra Huawei Türkiye AR-GE Merkezi içerisinde bulunan takımlar arasındaki bütünlüğü de olumsuz yönde etkilediği sonucuna varılmıştır. Kalite ve Operasyon Departmanı yapılan gözlemler ve elde edilen geri dönüşler sonucunda bir etkinlik düzenlemeye karar vermiştir. Oyun tasarımına katkıda bulunması açısından çalışanların bir kısmı ile yarı yapılandırılmış görüşmeler gerçekleştirilmiştir. Yapılan görüşmelerde, takımlar arasındaki etkileşimin az olduğu, çalışan bireylerin iş birliğinden çok kişisel çalışmaya yatkın olduğu ve aralarındaki iletişimin bilgi paylaşımı için yeterli olmadığı gözlemlenmiş ve çalışanların farklı departmanlardaki kişilerle ortak çalışma fırsatı bulmak istedikleri, daha dinamik ortamlarda bir araya gelmek istedikleri tespit edilmiştir. Bu çalışmada da dijital olmayan bir oyun tasarlanarak, yazılım sektöründeki bir firmanın çalışanları arasındaki iletişimin artırılması, iş birliğinin iyileştirilmesi, motivasyonun yükseltilmesi, örgütsel başarının artırılması ve bilgi paylaşımının çoğaltılması hedeflenmektedir. Bu bildirinin ikinci bölümünde ta-

sarlanan İş Birliği Oyunu, oyun akışı ve kuralları, oyunun değerlendirilmesinden sonra bulgular tartışılacaktır. Makale sonuçlar ve gelecek çalışmalar kısmı ile son bulacaktır.

2 İş Birliği Oyunu

İş Birliği Oyunu Huawei Turkiye AR-GE Merkezi Kalite ve Operasyon Departmanı tarafından eğlendirici olduğu kadar öğretici olmasına da dikkat edilerek tasarlanmış bir oyundur. Bu oyunun temelinde iki adet oyun yer almaktadır. Bunlar; *Pub Quiz* ve *Risk* oyunlarıdır. Genellikle bir masa etrafında oturan grupların soruları cevaplamak için birlikte çalıştıkları bir etkinlik olan Pub Quiz oyunu bir bilgi yarışmasıdır. Rastgele oluşturulan her bir gruba soruları cevaplayabileceği bir kâğıt verilir. Oyuncular sorulara bir ekran üzerinden erişim sağlarlar ve cevaplarını onlara dağıtılan kağıtlara yazarlar. Katılımcılar tüm sorulara belirlenen sürede yanıt verdikten sonra her takımdan cevap kağıtları toplanır. Moderatör tarafından katılımcıların cevapları değerlendirilirken oyuncular ekrandan soruların yanıtlarını kontrol ederler. En çok doğru yanıt veren takım oyunu kazanmış olur. *Pub Quiz* oyununda amaç, eğlenirken öğrenmek ve en çok doğru cevabı vererek oyun sonundaki ödülü kazanmaktır. Risk oyunu, iki ila altı oyuncu ile oynanabilen bir strateji, diplomasi, çatışma ve fetih oyunudur. Standart versiyonda, altı kıtaya ayrılmış kırk iki bölgeye bölünmüş, siyasi bir Dünya haritası üzerinde oynanmaktadır. Risk oyununda amaç hedefi gerçekleştirerek dünyanın hakimi olabilmektir. Askeri birlikleri stratejik konumlara yerleştirerek yeni bölgeleri fethedip hedefe ulaşılabilir (bkz. Şekil 1).

İş Birliği Oyunu ise bu iki oyundan esinlenilerek temelinde bilgi yarışması bulundurmasının yanı sıra bölgelere ayrılmış özel bir dünya haritası üzerinde oynanan strateji ve fetih içeren bir oyun olarak tasarlanmıştır. Tasarladığımız oyunda Huawei, Kalite ve Süreçleri, Yazılım Mühendisliği ve Genel Kültür kategorilerinde sorular hazırlanarak, katılımcıların hem bu kategorilerdeki bilgileri ölçülmüş hem de takım olarak belirledikleri ortak payda ve hedefler için stratejik kararlar almaları sağlanmıştır. İş Birliği Oyunu, Huawei Türkiye AR-GE Merkezi bünyesinde bulunan farklı departmanlardaki takımlar arası iletişimi, bilgi paylaşımını, bireylerin ortak payda ve hedefler için farklı durum ve koşul olasılıklarını düşünüp çözümler üreterek çalışmalarını, zamana karşı yarışmalarıyla verimli zaman yönetimi niteliklerini artırmayı amaçlamıştır. Tüm bunlar öğrenme, kendini geliştirme, eğlence, bir gruba bağlı olduğunu hissetme ve statü kazanma gibi iç motivasyonu tetikleyici elementlerle oluşturulmuştur.

İş Birliği Oyunu, belirli kategorilerde hazırlanmış sorulardan, pul, zar, piyon, bölge kartları, kıta gibi materyallerden ve dünya haritasından oluşmaktadır. Oyuna başlarken bölge kartları takımlara eşit olacak şekilde rastgele dağıtılır. Kalan bölge kartları, hazırlanan sorulardan takımlara 3 adet soru sorularak doğru ve en hızlı cevabı veren takıma verilmiştir. Takımlara sahip oldukları bölgeler doğrultusunda piyon ve pul materyalleri sağlanır. Oyuna birinci olarak başlayacak takım zar atılarak belirlenir ve zar sıralamasıyla devam edilir. Sırası gelen takım stratejisi gereğince elindeki bölgeleri savunmak ve yeni bölge ya



Şekil 1. Oyuncuların piyonlarını harita üzerine yerleştirmesi

da kıtalar kazanarak gücünü artırmak için savaş başlatma hakkına sahiptir. Savaşı sorulara doğru ve hızlı yanıt veren kazanır. Oyun belirlenen zaman dolunca biter. Oyun süresi bittiğinde doğru strateji yaparak en çok materyal, bölge ve kıtaya sahip olan takım oyunu kazanır. Puanlama sırasında 1 pul=1, 1 piyon= 2, 1 bölge=6, Afrika Kıtası=3, Asya Kıtası=7, Avustralya Kıtası=2, Avrupa Kıtası=5, Kuzey Amerika Kıtası=5, Güney Amerika Kıtası=2 puan değerinde hesaplanır.

2.1 Oyunun Hazırlık Aşaması

Oyunun hazırlık çalışmaları çeşitli aşamalardan oluşmaktadır. Öncelikle 225 sorudan oluşan bir soru havuzu oluşturulmuştur. Huawei Türkiye AR-GE Merkezi Kalite ve Operasyon Departmanı bünyesinde çalışmakta olan Kalite Güvence Mühendisleri ve Konfigürasyon Mühendisleri soruların hazırlanmasında görev almıştır. Soru havuzunun oluşturulması için, *Aşağıdakilerden hangisi regression test case değildir?* , *Huawei Enterprise Bulut Çözümleri arasında aşağıdaki teknolojilerden hangisi yer almaz?* gibi farklı kategorilerden oluşan sorular hazırlanmıştır. Mühendisler belirlenen kategoriler hakkında çeşitli sorular hazırlarken eş zamanlı olarak oyunun oynanması için gerekli olan materyaller hazırlanmıştır. Materyaller içerisinden öncelikle yüksek çözünürlüklü ve üzerine mıknatıslı piyonların tutturulabileceği 6 kıtaya (Kuzey Amerika, Güney Amerika, Avrupa, Afrika, Asya, Avustralya Takımadaları) bölünmüş dünya haritası örneği bulunup, duvara sabitlenmeye hazır bir şekilde bastırılmıştır (bkz. Şekil 2).



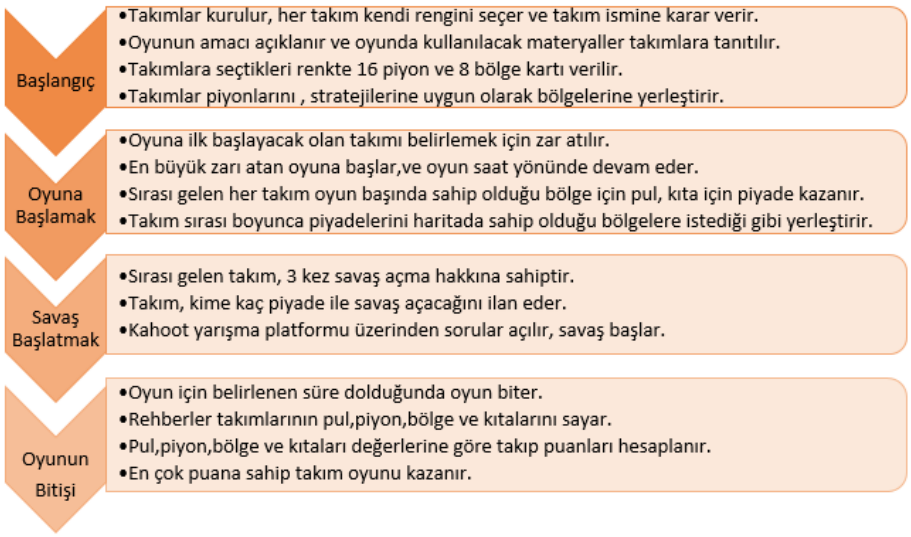
Şekil 2. İş Birliği Oyunu Haritası

Haritanın ve piyonların yanı sıra 42 adet bölge kartı hazırlanmıştır. Materyallerin hazırlanması devam ederken, soruların tamamlanmasının ardından oyun esnasında oyunculara soruları sormayı sağlayacak olan Kahoot isimli yarışma platformuna 2'li, 4'lü, 6'lı gruplar halinde sorular yüklenmiştir. Sorular ve materyaller hazırlanırken aynı zamanda katılımcıları belirlemek adına Kalite ve Operasyon Departmanı tarafından tüm çalışanlara katılma taleplerini bildirebilecekleri bir elektronik posta gönderilmiştir. Bu elektronik postaya iliştirilmiş olan bağlantıya tıklayıp katılmak istediğini belirten ilk 15 kişi oyuna katılımcı olmaya hak kazanmıştır. Esas oyunculara ek olarak sıralamaya göre yedek oyuncular da belirlenmiştir. Oyuncular farklı departmanlardan kişilerin bir araya getirilmesi esas alınarak farklı takımlara yerleştirilmiştir. İş Birliği Oyunu için gerekli olan her şey hazır olduktan sonra, oyunun oynanmasından önce farklı takımlardaki kişiler arasında bir bağ oluşmasına öncülük etmesi adına küçük ve kısa bir ortam ısıtıcı (ice breaker) oyun oynatılması planlanmıştır. Bu ortam ısıtıcı oyun için çeşitli araştırmalar yapıldıktan sonra İş Birliği Oyunu başlamadan hemen önce her ekip için bir kez sessiz sinema oynatılmasına karar verilmiştir.

2.2 Oyun Akışı ve Kuralları

Tasarladığımız İş Birliği Oyunu'nun ilk oturumu, oyun öncesi yapılan ve kişilerin katılım durumları ile rollerini belirtebilecekleri bir anket doğrultusunda, oyuna katılacak kişiler arasından departmanlar arası iletişimi artırmak amacı ile her biri farklı departmanlardan olacak sekilde üçer kişilik beş adet takım oluşturularak başlamıştır. Takımlarda en az bir yazılım mühendisi ve bir test mühendisi olmasına dikkat edilmiştir. Her takıma, oyun boyunca kuralları hatırlatmak ve materyalleri sağlamaktan sorumlu bir rehber atanmıştır. Rehberler Katile Güvence Mühendisleri'nden ve Konfigürasyon Mühendisleri'nden oluşmaktadır. Oyunda

kullanılacak materyaller (kartlar, pullar, piyonlar, zarlar, Kahoot yarışma platformu), harita takımlara tanıtılmış, oyunun amacı ve kuralları açıklanmıştır. Her takımdan kendini temsil edecek takım isimlerini belirlemeleri ve 5 ana renkten birini seçmeleri istenmiştir. Takımlara seçtikleri renkte oyun boyunca kullanacakları 16 adet piyon ve 8 adet bölge kartı verilmiştir. Takımlardan, takım içinde ortak karar alarak kendi stratejilerini belirlemeleri ve sahip oldukları piyonları bölge kartlarına stratejileri doğrultusunda yerleştirmeleri istenmiştir. Oyun, ilk başlayan takımdan itibaren saat yönünde devam etmiştir ve takımların oyun boyunca piyon, pul, bölge ve kıta kazanımları oyun kurallarında belirtildiği şekilde açılan savaşlarla sağlanmıştır. Takımlar arası savaşın kazananı Kahoot yarışma platformu üzerinden belirlenmiştir. Oyun belirtilen kurallarla üç saat boyunca sırasıyla devam etmiştir. Oyun süresi dolduğunda rehberler takımlarının pul, piyon, bölge ve kıtalarını saymıştır. Takımların sahip oldukları pul, piyon, bölge ve kıta için belirlenen değerlerde puanları hesaplanmıştır. En yüksek puana sahip takım oyunu kazanmıştır (bkz. Şekil 3).

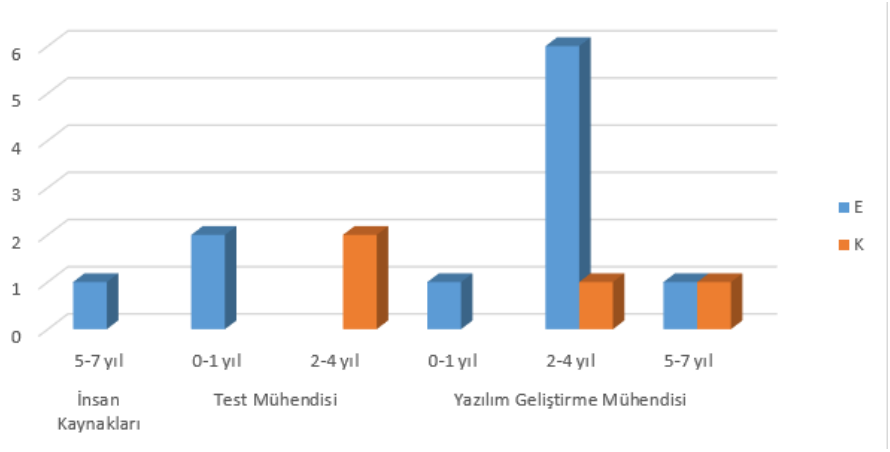


Şekil 3. İş Birliği Oyunu Akış Şeması

3 Bulgular ve Tartışmalar

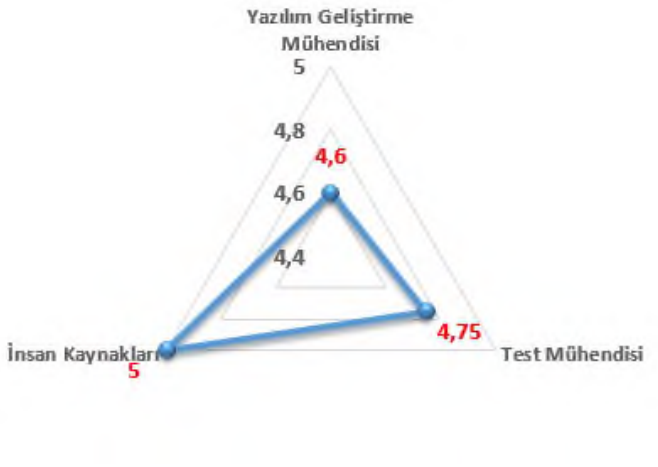
İş Birliği Oyunu'nun ardından, oyunun katılımcılar üzerindeki etkisini ölçebilmek için Huawei Türkiye AR-GE Merkezi Kalite ve Operasyon Departmanı tarafından bir anket hazırlanmıştır. Bu anket sonuçları değerlendirildiğinde katılımcıların %67'sinin Yazılım Geliştirme Mühendisi olduğu gözlemlenmiştir. Şekil 4'de

görüldüğü üzere Yazılım Geliştirme Mühendisleri'nin çoğunluğu 2-4 yıl arası toplam tecrübeye sahipken, test mühendisleri 0-1 yıl ve 2-4 yıl olarak eşit bir dağılıma sahiptir. İnsan Kaynakları Departmanı'ndan 5-7 yıl arası toplam tecrübeye sahip 1 kişi katılım göstermiştir. Katılımcıların rol dağılımları göz önüne alınmadan toplam tecrübe dağılımları incelendiğinde ise daha çok 2-4 yıl arası tecrübeye sahip kişilerin katılım sağladığı sonucuna varılmıştır. Aynı zamanda anket sonuçları incelenirken katılımcıların çoğunun erkek bireyler olduğu fark edilmiştir (bkz. Şekil 4).



Şekil 4. Katılımcıların Toplam Tecrübe ve Rol Dağılımları

Oyun sonrası hazırlanan ankette katılımcılar arasındaki iş birliğinin artıp artmadığının öğrenilmesini sağlayacak sorular yer almıştır. Örneğin; "Oyun, diğer ekip üyeleriyle iş birliğini artırmam için iyi bir yoldu." sorusu, bu sorulardan biri olarak hazırlanmıştır. İşbirliği ile ilgili bu soruların sonucunda katılımcılardan yazılım geliştirme mühendisi olanlar oyunun işbirliklerine katkılarını 5 üzerinden ortalama 4,6 olarak değerlendirmişlerdir. Test mühendisi olan katılımcılar ise 5 üzerinden ortalama 4,75 olarak değerlendirirken, insan kaynakları çalışanı olan katılımcı 5 üzerinden 5 tam puan vermiştir (Şekil 5). Huawei Türkiye AR-GE Merkezi Kalite ve Operasyon Departmanı olarak bu oyunu tasarlamaktaki amaç çalışanlar arasındaki işbirliği yeteneğinin artırılmasıdır. Aşağıdaki grafiğe göre katılımcıların geri dönüşlerini değerlendirecek olursak İş Birliği Oyunu'nun katılımcılar arasındaki iş birliğini, iletişimi ve paylaşımı son derece iyi yönde etkilediği görülmektedir.



Şekil 5. Rol Dağılımına Göre İşbirliği Artışı

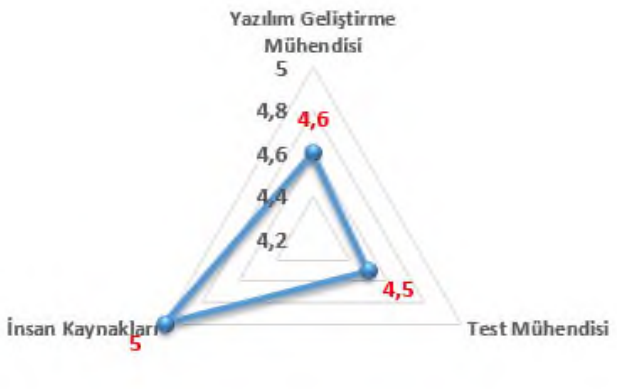
Hazırlanan ankette, kişilerden oynanan oyunun bilgi paylaşımına ve zaman yönetimine sağladığı katkıyı puanlaması da istenmiştir. 5 üzerinden yapılan bu puanlandırmada yazılım geliştirme mühendisleri bu sorulara ortalama olarak 4,5 puan ve insan kaynakları çalışanı 4 puan verirken, test mühendislerinin 4,25 puan verdiği Şekil 6'da görülmektedir. Oyun öncesi yapılan gözlemlerde işbirliği eksikliğinin sonuçları olarak bilgi paylaşımı ve zaman yönetiminin de verimli olarak sağlanamadığı fark edilmiştir.

Aşağıdaki grafiğe göre oynanan İş Birliği Oyunu, oyuncuların bilgi paylaşımını olumlu yönde etkilemiştir ve zamanı daha iyi yönetmelerini sağlamıştır. Kalite ekibi yapılan anketin yanısıra oyun sonrasında da katılımcıları izlemeye devam etmiş ve oyunun faydalarını iş yaşantılarına taşıyıp taşımadıklarını gözlemleme fırsatı bulmuştur. Yapılan gözlemler ve ortak çalışmalar sonucunda bilgi paylaşımı ve zaman yönetiminin oyun öncesine göre daha başarılı bir şekilde ilerlemekte olduğu fikrine varılmıştır.



Şekil 6. Rol Dağılımına Göre Bilgi Paylaşımı Artışı ve Zaman Yönetimine Sağlanan Katkı

Ankette katılımcıların İş Birliği Oyunu esnasında yeni kişilerle tanışma oranını, oyun sonrasındaki iletişimlerini ve arkadaşlık kazanma oranını puanlayan sorular da sorulmuştur. Bu soruların yanıtlarından çıkan sonuçlarda Yazılım Geliştirme Mühendisi olan katılımcılar ortalama olarak 5 üzerinden 4,6 puan vermiştir. Test Mühendisi olan katılımcılar 5 üzerinden ortalama olarak 4,5 vermiştir. İnsan kaynakları çalışanı ise 5 üzerinden 5 tam puan vermiştir (bkz. Şekil 7). Katılımcıların sosyalleşmesi olarak değerlendirdiğimiz bu sonuçlar göstermektedir ki, gerçekleştirilen iyileştirmeler sadece oyun esnasına değil oyun sonrasındaki iş yaşantısına da tesir etmiştir. Katılımcıların oyun sonrasındaki iletişimlerinde meydana gelen bu iyileşmeler, İş Birliği Oyunu'nun esas tasarlanma amacı olan iş birliğini artırma hedefini doğrudan etkileyen önemli faktörlerdir. Bu nedenle katılımcıların sosyalleşme artışı konusundaki değerlendirmeleri oyunun amacına ulaştığının bir göstergesi olarak değerlendirilebilir.



Şekil 7. Rol Dağılımına Göre Katılımcılardaki Sosyallik Artışı

4 Sonuçlar ve Gelecek Çalışmalar

Bu çalışma Huawei Türkiye AR-GE Merkezi bünyesindeki takımlar arasında gözlemlenen iş birliği, bilgi paylaşımı, iletişim ve motivasyon eksikliklerinin oyunlaştırma ile giderilmesi amacıyla yapılmıştır. Yapılan bu organizasyon ilk aşamada başvuran 15 kişi ile oynanmış olup, gelecek organizasyonlarda daha çok kişiye ulaşılması amaçlanmaktadır. Bu çalışmada iş birliği artışının takımlar arası etkileşime olan katkısı, yaptığımız anketler vasıtası ile katılımcılar tarafından da doğrulanmıştır. Yapılan anketler sonucunda oyuncular, İş Birliği Oyunu'nun genel başarısını 5 üzerinden ortalama 4.65 olarak puanlamıştır. Bu ortalama bize İş Birliği Oyunu'nun %93 oranında başarıya ulaştığını göstermektedir.

Tasarladığımız İş Birliği Oyunu'nun daha verimli olabilmesi ve daha büyük kitlelere ulaşabilmesi için oyunun farklı takımlar üzerinde tekrarlanması, soru havuzunun güncellenmesi ve çeşitlendirilmesi gerekmektedir. Bunların yanı sıra, oyun öncesi ve sonrası arasındaki iş birliği artışının daha iyi analiz edilebilmesi için, oyun öncesinde yapılan röportajlara ek olarak anket yapılması gerekmektedir. Gelecek hedeflerimiz arasında, İş Birliği Oyunu ile birlikte takımların ortak payda ve hedefler için çalışmalarına katkıda bulunmak ve oyun kültürünün benimsenmesini sağlamak yer almaktadır.

Kaynakça

1. Danış, S.: An analysis of gamers' satisfaction levels: The dualities of digital and non-digital games. Master's thesis, Izmir Institute of Technology (2015)
2. Yilmaz, M., O'Connor, R.V.: Maximizing the value of the software development process by game theoretic analysis. In: 11th International Conference on Product

Focused Software (PROFES 2010). PROFES '10, New York, NY, USA, ACM (2010) 93–96
3. Adams, E.: Fundamentals of game design. Pearson Education (2014)
4. Yilmaz, M., O'Connor, R.: Yazilim gelistiren organizasyonlar icin dusunulmus bir kisilik tipi olcum oyunu. In: 7th Turkish National Software Engineering Conference (UYMS 2013). (2013) 1–12
5. Yilmaz, M., O'Connor, R.V.: Oyun kuramı kullanarak yazılım takımlarının uretkenligini artırmak için gelistirilen bir yazılım sureç muhendisligi yaklasımı. In: 5th Turkish National Software Engineering Symposium (UYMS). (2011) 5357–5367
6. Huizinga, J.: Homo Ludens Ils 86. Routledge (2014)
7. Camerer, C.F.: Behavioural studies of strategic thinking in games. Trends in cognitive sciences **7** (2003) 225–231
8. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.V.: Algorithmic game theory. Cambridge university press (2007)
9. Brams, S.J.: Game theory and the humanities: bridging two worlds. MIT press (2011)
10. Kaymak, M., Namıduru, M., Tuzun, E., Yilmaz, M.: Sanal ofis ortaminda kod gozden gecirme ile kod degerlendirmesi. In: 10th Turkish National Software Engineering Conference (UYMS 2016). (2016) 706–711
11. Gulec, U., Gozcu, M., Dogan, S., Mesurhan, N., Yilmaz, M., Isler, V., Dinc, M.: Simulacrum: Savas kosullarinda acil tibbi mudahale ve ilk yardim simulasyonu. In: 12th Turkish National Software Engineering Conference (UYMS 2018). (2018) 1–11
12. Yilmaz, M., O'Connor, R.: A scrumban integrated gamification approach to guide software process improvement: a turkish case study. Tehnicki Vjesnik (Technical Gazette) **23** (2016) 237–245
13. Kosa, M., Yilmaz, M., O'Connor, R., Clarke, P.: Software engineering education and games: A systematic literature review. Journal of Universal Computer Science **22** (2016) 1558–1574
14. Baker, A., Navarro, E.O., Van Der Hoek, A.: An experimental card game for teaching software engineering processes. Journal of Systems and Software **75** (2005) 3–16
15. Yilmaz, M.: A software process engineering approach to understanding software productivity and team personality characteristics: an empirical investigation. PhD thesis, Dublin City University (2013)

ERP Projelerinde Efor Kestirimi

Neslihan Küçükateş Ömüral[1] and Onur Demirörs[2]

[1] Enformatik Enstitüsü, Orta Doğu Teknik Üniversitesi, Ankara, Türkiye
[2] Bilgisayar Mühendisliği, İzmir Yüksek Teknoloji Üniversitesi, İzmir, Türkiye
[1] kucukates@gmail.com [2]onurdemirors@iyte.edu.tr

Özet. ERP sistemleri işletmelerin temel süreçlerinin çoğunu kapsayan, entegre ve karmaşıklık seviyesi yüksek sistemlerdir. ERP projelerinin en kritik aşamalarından biri de efor kestirimidir. Bu tarz projelerde efor kestiriminin doğru yapılamaması yüzünden; projelerin çoğu, zamanında ve belirlenen bütçe içerisinde bitmemektedir. Yazılım geliştirme projeleri için birçok efor kestirim çalışması olmasına karşın, ERP projeleri için bu alanda yapılmış çalışma sayısı kısıtlıdır. Bu bildiride, ERP projelerini efor kestirim alanında farklılaştıran noktalar ve uygun efor kestirimi için kullanılması gereken parametreler, bu alanda yaptığımız çalışmalarda edindiğimiz tecrübeler ışığında anlatılmıştır.

Anahtar Kelimeler: ERP, Efor Kestirimi, Yeniden Kullanım, Değişiklik.

Effort Estimation in ERP Projects

Neslihan Küçükateş Ömüral[1] and Onur Demirörs[2]

[1] Informatics Institute, Middle East Technical University, Ankara, Turkey
[2] Software Engineering, İzmir Institute of Technology, İzmir, Türkiye
[1] kucukates@gmail.com [2]onurdemirors@iyte.edu.tr

Abstract. ERP systems are complicated and integrated systems, covering most of the business processes of an enterprise. Effort estimation is one of the most critical phases of ERP projects. These projects suffer from time and budget overruns due to improper effort estimation. Although there are many studies for effort estimation of software development projects, there are limited number of studies in this field for ERP projects. In this paper, the points that differentiate ERP projects in the effort estimation field and the parameters that should be used for appropriate effort estimation are explained in the light of our experiences we gained in our studies.

Keywords: ERP, Effort Estimation, Reuse, Change.

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

1 Giriş

ERP (Kurumsal Kaynak Planlaması) sistemleri bir kurumun süreçlerinin çoğunu kapsayan, büyük ve karmaşık sistemlerdir. ERP projeleri uyarlama, entegrasyon, veri transferi gibi bu projelere özel süreçler ile klasik yazılım geliştirme projelerinden ayrılır. Bu tarz projelerin efor kestirimi için özel olarak geliştirilmiş ve genel kabul gören bir efor kestirim yöntemi bulunmamaktadır [1]. Bu tarz projelerde efor kestiriminin doğru yapılamaması yüzünden, proje takvimi ve bütçesinde büyük sapmalarla karşılaşılmaktadır. [16]

Daneva ve Wieringa'nın [4] çalışmalarında belirttiği gibi, ERP projeleri diğer yazılım projelerinden farklı değerlendirilmelidir, bu projeler binlerce iş sürecini kapsar, muhtelif konfigürasyon ve değişiklik ihtiyaçları vardır. Bu karmaşıklığının yanında, tüm yapıyı tasarlayan tek bir tasarımcı yoktur, her parçası ayrı ekipler tarafından çalışılır. Yapılan çalışmalarda[5] ERP uygulama maliyetlerinin beklenenden yüksek gerçekleştiği, bunun yanı sıra öngörülmeyen eğitim, uyarlama, entegrasyon, modifikasyon gibi maliyet kalemlerinin ortaya çıktığı görülmüştür.

ERP projelerinin yaşam döngüsü şelale yöntemine benzerlik gösterir. Proje hazırlığı safhasında "Kapsam" dokümanı, kavramsal tasarım aşamasında da gereksinimlerin ve konfigürasyon ihtiyaçlarının detaylandırıldığı "Kavramsal Tasarım" dokümanları hazırlanır. Kapsam dokümanında devreye alınacak modüller ve senaryolar; kavramsal tasarım dokümanında da uygulanacak iş süreçleri ile ilgili detaylar anlatılır. Bu dokümanlar efor kestirim yöntemleri için kullanılabilecek temel kaynak dokümanlardır.



Şekil 1. ERP Projelerinde Proje Yaşam Döngüsü

Bu bildirinin amacı ERP projelerinde efor kestirim yöntemlerini incelemek, bu yöntemlerin faydaları ve zorluklarını, bu yöntemlerde kullanılması gereken parametreleri tartışmak, bu yöntemlerin ERP projelerinin gereksinimleri doğrultusunda iyileştirilmesi konusunda yol gösterici olmaktır.

Bildirinin geri kalanı şu şekilde düzenlenmiştir: 2. bölümde ERP projelerinde efor kestirimi alanında yapılmış, literatürde geçen araştırmalar, 3. bölümde bizim bu alanda yaptığımız çalışmalar ve çıkarımlarımız, 4. bölümde ise sonuç ve gelecek çalışmalar anlatılmıştır.

2 Yapılan Araştırmalar

Bu alanda yapılan ilk araştırma [6], Stensrud ve Myrtveit'ın geleneksel efor kestirim yöntemlerinin ERP gibi alanlarda kullanımının uygunluğunun tartışıldığı çalışmalarıydı. Bu çalışmada, ERP projelerinin maliyeti etkileyen birçok farklı parametresi olduğunu, efor kestiriminde bu parametrelerinin dikkate alınması gerektiğini tespit ettiler. Bu parametreler 48 adet proje seti için analoji aracı ANGEL ve çoklu regresyon analizi yöntemleri ile denendi ve doğrulandı.

2001 yılında Stensrud [7] araştırmasında, ERP projelerinin efor kestiriminin, sadece fonksiyon nokta veya LOC (Kod Satır Sayısı) kullanılarak değil, çok boyutlu proje büyüklüğü ölçütlerinin de kullanılarak yapılması gerektiğini tespit etti. Bu çalışmada, ERP projelerine uygun büyüklük ölçütleri analiz edildi. Olası ERP büyüklük ölçütleri; modül, kullanıcı, yazılım arayüzü, lokasyon, işletme birimleri, EDI arayüzü, veri dönüşümleri, özel geliştirilmiş raporlar ve değiştirilmiş edilmiş programlar olarak belirlendi. Ayrıca, yine bu çalışmada mevcut efor kestirim modellerinin ERP projelerine uygunluğu da analiz edildi. Stensrud'un araştırması gibi, mevcut efor kestirim modellerinin ERP projelerine uygunluğunu inceleyen farklı araştırmalar [14,15] yapıldı.

ERP alanında fonksiyon nokta analizlerini temel alan araştırmalar da yapıldı [1]. Tellez [8] COSMIC fonksiyon nokta analizi ve EPC (Olay Tabanlı Süreç Zinciri) diyagramlarını kullanarak, ERP iş süreçleri bazında büyüklük ölçümü için bir yöntem geliştirdi ve bunu bir SAP projesi üzerinde uyguladı. Kujipers [9], araştırmasında IFPUG fonksiyon noktalarını SAP projeleri için otomatik olarak hesaplayan bir yöntem önerisinde bulundu. ERP projelerinde fonksiyon nokta bazlı efor kestirimi ile ilgili üç temel araştırma [10,11,12] yapıldı. Erasmus[10], ERP kavramsal tasarım dokümanlarında belirtilen gereksinimleri inceleyerek iş süreçleri ve adımlarını listelediği, bu adımlar bazında COSMIC fonksiyon noktaları ölçtüğü ve dönüşüm faktörleri kullanarak, proje için efor kestirimi yaptığı "COSMIC EPC" yöntemini geliştirdi; bu yöntemi bir SAP projesi için uyguladı. Vogelazang [11], ERP projelerinde erken efor kestirimi için yine COSMIC fonksiyon noktaları kullanarak bir yöntem geliştirdi. Pierre ve Daneva [12], her durum için aynı efor kestirim yöntemini kullanmak yerine fonksiyon nokta ve uzman görüşü yöntemlerini entegre eden kestirim stratejilerini kullanmayı önerdi. Daneva [13], yeniden kullanım oranlarının ERP efor kestirimine etkisi ile ilgili bir araştırma gerçekleştirdi. Bu araştırma kapsamında fonksiyon noktalara dayalı yeniden kullanım oranlarının ölçülmesi üzerine bir yöntem tanımlandı. Bu yöntem ile yeniden kullanımı üç ana seviyede kategorize etti ve üç SAP projesi için bu yeniden kullanım seviyelerini hesapladı.

3 Çalışmalarımız ve Çıkarımlarımız

ERP projelerinde efor kestirim problemini analiz etmek, bu alanda yapılmış araştırmaları tespit etmek ve incelemek için bir sistematik literatür taraması gerçekleştirdik [1]. Literatür taraması ile ulaştığımız 41 araştırmanın büyük bölümü mevcut efor kestirim yöntemlerinin ERP projelerinde uygulanması ile ilgili çalışmalardı. 10 adet çalışmada yeni efor kestirim yöntemleri öneriliyordu, bu yöntemlerin çoğu da fonksiyon nokta bazlı büyüklük ölçümünü temel alan yöntem önerileriydi.

Literatür taraması ile ulaştığımız araştırmaları incelediğimizde, "ERP projelerinde efor kestirimi"nin bu alanda hala güncel bir konu olduğunu; bu tür projelerin efor kestiriminin nasıl yapılacağı konusunda bir fikir birliğine varılamadığını tespit ettik. ERP sistemlerinin her yıl müşteri ve endüstri beklentilerine dayalı olarak gelişmekte, dönüşmekte olduğunu; bu nedenle, eski proje verilerinin yeni projelerin efor kestirimi için temel kaynak olarak kullanılmasının, benzerlik yoluyla efor kestirimi yapmanın ERP projeleri için çok doğru bir yöntem olmadığı sonucuna vardık. ERP projelerinde efor

kestiriminin düzgün bir şekilde yapılabilmesi için yeni projenin müşteri gereksinimlerinin baz alınması gerektiğini, bu noktada en doğru kaynağın ERP projelerinin kavramsal tasarım dokümanları olduğunu gördük. Kavramsal tasarım dokümanları fonksiyon nokta bazlı büyüklük ölçümleri için yaygın olarak kullanılıyordu. Bu alandaki çalışmaların temel eksiği ise, geliştirilen yöntemlerin geçerlemesinin farklı tip ve büyüklüklerde ERP projeleri ile yapılmamış olmasıydı.

Fonksiyon nokta bazlı büyüklük ölçümüne dayanan efor kestirim yöntemlerinin ERP projelerinde uygulanabilirliğini anlamak için, bu alanda yapılmış üç araştırmada [10,11,12] tarif edilen yöntemleri bir durum çalışmasında uyguladık [2]. Bu durum çalışması için, danışmanlık firmasından gerçekleşen efor verileri ve kavramsal tasarım dokümanlarını aldığımız bir SAP projesini kullandık. Uyguladığımız üç yöntemde de büyüklük ölçümü için COSMIC fonksiyon nokta kullanılıyordu. COSMIC fonksiyon nokta hesaplaması için gerekli olan veri hareketleri, kavramsal tasarım dokümanlarından çıkarılan iş süreçleri analiz edilerek belirleniyordu. Bu efor kestirim yöntemleri ile ulaştığımız efor kestirimleri için MRE (Ortalama Bağıl Hata) değerleri 0,39-0,53 aralığında gerçekleşti.

Bu durum çalışması ile, bu iş süreçlerinin efor kestirimi için değerli kaynaklar olduğunu tespit ettik. COSMIC fonksiyon nokta yaklaşımının, ERP projelerinin büyüklük ölçümü için uygun yöntemlerden biri olabileceğini değerlendirdik. Bu çalışma ile, fonksiyon nokta tabanlı ERP efor kestirim yöntemleri için kritik olan iki parametrenin; değişiklik ve yeniden kullanım seviyesi olduğunu tespit ettik. ERP projelerinde doğru kestirim yapabilmek için bu iki parametrenin net olarak tanımlanması ve ölçülmesinin kritik olduğu sonucuna vardık. Bu iki parametrenin yanı sıra, efor kestiriminde dönüşüm faktörleri kullanmak yerine, net ölçülmüş üretkenlik değerleri kullanmanın daha doğru sonuçlara ulaşmayı sağlayacağı çıkarımını yaptık.

Değişiklik ve yeniden kullanım seviyesinin yansıtıldığı bir efor kestirim metodunun uygulanabilirliğini anlamak için, bu alanda yaptığımız çalışmaları uyguladığımız ikinci bir durum çalışması gerçekleştirdik [3]. Bu durum çalışması için yine bir SAP projesini kullandık. Uyguladığımız yöntemde, proje büyüklüğünü COSMIC fonksiyon nokta olarak ölçtük, yeniden kullanım seviyelerini belirlemek için Daneva [13] tarafından ERP projeleri için önerilen seviyeleri kullandık.

Tablo 1. ERP Yeniden Kullanım Seviyeleri

Yeniden Kullanım Seviyesi	Yeniden Kullanım Tanımı
Seviye1	Süreç ve veri bileşenleri üzerinde büyük değişiklikler (kod seviyesinde değişiklik) yapılması
Seviye2	Süreç ve veri bileşenleri üzerinde küçük değişiklikler (akışı değiştirmeyen parametre değişiklikleri) yapılması
Seviye3	Süreç ve veri bileşenlerinin hiç değişiklik yapılmadan kullanılması
Yeniden kullanım yok	Yeni geliştirme

Bu yeniden kullanım seviyeleri için belirlediğimiz katsayıları kullanarak, yeniden kullanım seviyesi yansıtılmış yeni büyüklük değerine ulaştık. COSMIC ile 247 fonksi-

yon nokta olarak ölçtüğümüz proje büyüklüğünü, yeniden kullanım seviyelerini yansıtarak yaptığımız bu ölçümle 112,55 COSMIC fonksiyon nokta olarak hesapladık. Durum çalışmamızda ulaştığımız bu büyüklük değerinin geçerli olup olmadığını anlamak için, bu değeri mevcut efor kestirim yöntemlerinden biri [11] içerisinde kullandık. Ulaştığımız efor kestirim değeri, yeniden kullanım seviyeleri yansıtılmamış büyüklükle ulaşılan efor kestirim değerine göre çok daha iyi bir sonuç verdi. Daha önce 0,39 olarak hesaplanan MRE değeri, yeni ölçümle 0,19 olarak hesaplandı.

Bu çalışmamızda da ERP kavramsal tasarım dokümanlarının, fonksiyon nokta analizi için uygun bir kaynak olduğunu, bu dokümanların analizi ve iş süreçleri kütüphanelerinin kullanılması ile COSMIC fonksiyon noktalarının hesaplanabildiğini gördük. ERP kavramsal tasarım dokümanları, ilgili iş süreçleri için küçük veya büyük geliştirmelerin gerekli olup olmadığını anlamak için gerekli tüm detayları içeriyordu. Yeniden kullanım seviyesi tanımları bu geliştirmeleri ayırt etmek için yeterliydi. Yaklaşımımızda kullanılan yeniden kullanım seviyeleri, gereksinimlerin sınıflandırılabilmesi için faydalıydı; ancak, yeniden kullanma seviyeleri ve sabit katsayılar kullanmak yerine, ERP projelerinde yeniden kullanma oranını net olarak ölçmenin mümkün olup olmadığının analiz edilmesi gerektiği düşünüyoruz.

4 Sonuç ve Gelecek Çalışmalar

ERP projelerinde efor kestiriminin nasıl yapılması gerektiğini anlamak için bu alanda sistematik literatür taraması, proje analizleri ve durum çalışmaları gerçekleştirdik. Bu çalışmalar ile ERP projelerinin standart yazılım projelerinden farklı olduğunu, bu tarz projelere özel efor kestirim yöntemleri geliştirmek gerektiği sonucuna vardık.

ERP sistemleri hızla dönüşüp geliştiğinden, bu alanda geliştirilecek efor kestirim yöntemi, yeni bir kullanıcı tarafından bile objektif olarak uygulanabilir basitlikte olmalıdır. ERP projelerinin gereksinim dokümanı olan “Kavramsal Tasarım Dokümanı" efor kestirim yöntemi için temel kaynak olarak kullanılabilir. SAP Solution Manager gibi ERP proje araçları kavramsal tasarım dokümanının oluşturulması için de kullanılabiliyor. Kavramsal tasarım dokümanını temel alan büyüklük ölçüm ve efor kestirim yöntemlerinin bu araçlar üzerinde kullanılması ile hesaplamalar manuel işleme gerek kalmadan araç üzerinde otomatik olarak yapılabilir.

ERP projelerini farklılaştıran temel unsur yeniden kullanım oranlarının yüksek olmasıdır. Bu projelerin büyüklük ölçümleri yeniden kullanım oranları dikkate alınmadan yapıldığında sadece efor kestirimi değil, bütçeleme, proje planı gibi aşamaları da olumsuz etkilemektedir. Yeniden kullanımın ne ölçüde yapıldığı net olarak ölçülmeli ve bunun yansıtıldığı büyüklük değerleri üzerinden kestirim yapılmalıdır.

Gelecek çalışmalarımızda ERP projelerinde yeniden kullanımı dikkate alan bir büyüklük ölçüm yöntemi bulup, bu yöntemle gereksinimleri inceleyerek ERP’nin sağladığı standart çözümler üzerinde ne kadar değişiklik yapılması gerektiğini, ne kadar yeni geliştirme yapılması gerektiğini net olarak ölçmeyi hedefliyoruz. Bu ölçümlere dayanan bir büyüklük değeri üzerinden yapılacak efor kestiriminin çok daha doğru sonuçlar vereceğine inanıyoruz.

Kaynakça

1. Küçükateş Ömüral, N., & Demirörs, O. (2017, August). Effort Estimation for ERP Projects—A Systematic Review. In 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA) (pp. 96-103). IEEE.
2. Küçükateş Ömüral, N., & Demirörs, O. (2017, October). Effort estimation methods for ERP projects based on function points: a case study. In Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement (pp. 199-206). ACM.
3. Demirörs, O., & Küçükateş Ömüral, N. (2018, August). Exploring Reuse Levels in ERP Projects in Search of an Effort Estimation Approach. In 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA) (pp. 191-197). IEEE.
4. Daneva, M., & Wieringa, R. (2008). Cost estimation for cross-organizational ERP projects: research perspectives. Software Quality Journal, 16(3), 459-481.
5. Singh, A., & Wesson, J. (2009, October). Evaluation criteria for assessing the usability of ERP systems. In Proceedings of the 2009 annual research conference of the South African Institute of Computer Scientists and Information Technologists (pp. 87-95). ACM.
6. Stensrud, E., & Myrtveit, I. (1998, November). Human performance estimating with analogy and regression models: an empirical validation. In Proceedings Fifth International Software Metrics Symposium. Metrics (Cat. No. 98TB100262) (pp. 205-213). IEEE.
7. Stensrud, E. (2001). Alternative approaches to effort prediction of ERP projects. Information and software technology, 43(7), 413-423.
8. Téllez, F. M., Daneva, M., & Condori-Fernandez, N. (2009). Solving the size estimation problem in ERP project context: The eEPC-COSMIC approach (Doctoral dissertation, Master thesis, University of Twente, Enschede, The Netherlands).
9. Kuijpers, C. (2014, October). Automated FPA (eFPA) in SAP Environment-Visions and Experiences of Automated Function Point Analysis. In 2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (pp. 72-78). IEEE.
10. Erasmus, P. I. (2012). The COSMIC EPC method-An ERP functional size measurement method delivering time and cost estimates.
11. Vogelezang, F. (2006, November). Using COSMIC-FFP for sizing, estimating and planning in an ERP environment. In Proceedings of the 16th International Workshop on Software Measurement (IWSM 2006).
12. Erasmus, P., & Daneva, M. (2015, March). ERP Services Effort Estimation Strategies Based on Early Requirements. In REFSQ Workshops (Vol. 2015, pp. 83-99).
13. Daneva, M. (2014, October). Understanding functional reuse of ERP requirements in the telecommunication sector: an empirical study. In 2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (pp. 216-221). IEEE.
14. Daneva, M., & Wieringa, R. (2008). Cost estimation for cross-organizational ERP projects: research perspectives. Software Quality Journal, 16(3), 459-481.
15. Daneva, M. (2007). Preliminary results in a multi-site empirical study on cross-organizational ERP size and effort estimation. In Software Process and Product Measurement (pp. 60-71). Springer, Berlin, Heidelberg.
16. Panoroma, https://www.panorama-consulting.com/what-does-our-2019-erp-report-reveal-about-the-erp-industry/, last accessed 2019/06/15.

Türkiye'deki Yazılım Geliştirme İşleri İçin İstenen Becerilerin Metin Madenciliği Kullanılarak Analiz Edilmesi

Görkem Giray [0000-0002-7023-9469]

Bağımsız Araştırmacı, İzmir, Türkiye
gorkemgiray@gmail.com

Özet. Yazılım endüstrisinde istenen becerileri anlamak, üniversite müfredatı tasarlamak, eğitimler düzenlemek, çevrimiçi kurslar başlatmak, yazılım geliştiricilere kendini geliştirme konusunda rehberlik etmek gibi birçok konuda kritik öneme sahiptir. Bu amaçla, Türk yazılım geliştirme endüstrisinde en çok istenen teknik ve sosyal becerileri bulmak için kariyer.net'den 1.597 iş ilanını taradık ve analiz ettik. Analizimiz, dünya çapındaki son eğilimlerle tutarlı olarak, SQL, JavaScript ve HTML/CSS dillerinde deneyim için önemli bir talep ortaya koymaktadır. ASP.NET ve MS SQL Server, veri kümemize göre iş ilanlarında bilgisine en çok ihtiyaç duyulan baskın web çerçevesi ve veritabanıdır. Linux işletim sistemi hakkında bilgi, StackOverflow yazılım geliştirici anketi 2019 sonuçlarıyla tutarlı olarak, en çok istenen beceridir. Visual Studio, en çok istenen geliştirme ortamı ve .NET en baskın çerçevedir. Veri kümemize göre, işverenler tarafından yalnızca birkaç yazılım test aracı için deneyim aranmaktadır. En çok istenen sosyal beceriler takım çalışması, iletişim ve analitik/problem çözme becerileridir.

Anahtar Kelimeler: Teknik Beceri, Sosyal Beceri, Türk Yazılım Geliştirme Endüstrisi, Metin Madenciliği, Konu Modelleme.

An Analysis of Desired Skills for Software Development Jobs in Turkey Using Text Mining

Görkem Giray [0000-0002-7023-9469]

Independent Researcher, Izmir, Turkey
gorkemgiray@gmail.com

Abstract. Understanding the desired skills in software industry is critical in many aspects, including designing university curricula, organizing trainings, launching online courses, guiding software developers for self-development. To this end, we crawled and analyzed 1,597 job ads from kariyer.net to find most

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

desired technical and soft skills in Turkish software development industry. Our analysis reveals a substantial demand for experience in SQL, JavaScript and HTML/CSS languages in line with the recent trends worldwide. ASP.NET and MS SQL Server are the dominant web framework and database whose knowledge is needed most in job ads according to our dataset. Knowledge on Linux operating system is the most desired skill consistent with StackOverflow developer survey 2019 results. Visual Studio is the most desired development environment and .NET is the dominant framework. According to our dataset, experience on only a few software testing tools are sought by employers. The most desired soft skills are team work, communication and analytical/problem solving skills.

Keywords: Technical Skill, Soft Skill, Turkish Software Development Industry, Text Mining, Topic Modeling.

1 Introduction

Understanding the desired skills in software industry is critical in many aspects, including designing university curricula, organizing trainings, launching online courses, guiding software developers for self-development. To this end, we crawled and analyzed job ads from kariyer.net to answer the following research questions (RQs):

- RQ1: What are the most desired technical skills (experience on languages, web frameworks, databases, platforms, development environments, libraries, tools and software testing tools) in Turkish industry?
- RQ2: What are the most desired soft skills (non-technical skills) in Turkish software industry?
- RQ3: How can we classify job ads and analyze the various aspects of these clusters?

The rest of the paper is organized as follows: Section 2 presents the related work. Section 3 explains our research method in detail. In Section 4, we present our results and comment on these results. Section 5 includes the threads to validity. Finally, Section 6 concludes the paper.

2 Related Work

Table 1 displays the related work on analyzing desired skills by processing job ads. [1] has the biggest dataset by far compared to the other studies as well as our study. The rest of the studies have analyzed thousands of job ads and have a dataset of a similar scale with our study. Our study aims to analyze the desired skills for Turkish software industry and differs from the other studies in this respect. [2] presents a similar analysis for Thailand.

Table 1. Related work on analyzing desired skills by processing job ads.

Ref.	Year	Scope
[1]	2009	• crawled job ads from Monster.com, HotJobs.com, and SimplyHired.com daily between July 2007 and April 2008. • analyzed approximately 210,000 job ads.
[3]	2012	• crawled job ads from workopolis.ca (North America), eurojobs.com (Europe), monsterindia.com (Asia), and seek.com.au (Australia). • analyzed 500 ads for IT positions on the soft skills mentioned.
[4]	2017	• crawled job ads from stackoverflow.com. • analyzed 1,736 job ads.
[5]	2018	• gathered German information systems related job starter postings from four large online platforms: two general platforms (stepstone.de and monster.de) and two platforms specializing in entry-level jobs (get-in-it.de and absolventa.de). • analyzed 6,848 job ads.
[2]	2018	• crawled job ads (whose company location is Thailand) from two international job search engines, indeed.com and jobsdb.com. • analyzed 2,229 job ads.

Besides the similar studies in the literature, we used the results of two surveys [6], [7] in discussing our results.

3 Research Method

Fig. 1 displays the research method we applied in this study. In the first step, we crawled 2,914 job ads from kariyer.net, a web site for job and employee search in Turkey. We crawled the job ads categorized under the main category named "informatics and telecom". We included the sub-categories related to development, database/data warehouse, mobile programming, project management/business analysis, testing, product management and web-based application development. Even though we carefully selected the sub-categories to be included, there were many job ads that are irrelevant to software development and hence to our analysis. We manually filtered out the irrelevant job ads by checking job titles (step 3 in Fig. 1). There were many job ads that are not related to software development, such those seeking for networking experts, helpdesk technicians, etc. In addition, we also filtered out job ads whose working places are out of Turkey. After filtering out irrelevant job ads, we

ended up with a dataset consisting of 1,597 job ads. In the fourth step, we prepared our dataset for mining. We translated all job ads written in English to Turkish using Google Translation service.

In parallel with first, third and fourth steps, we also formed a list of skills to be searched in job ads (step 2 in Fig. 1). We used the items in StackOverflow's survey [6] conducted this year for languages, web frameworks, databases, platforms, development environments, libraries and tools. Since this survey does not include testing tools, we extracted a list of software testing tools from Wikipedia. We formed a unified list for soft skills by using the earlier studies in the literature [3], [4], [8].

In the fifth step, we mined the skills in job ads. We tried to minimize the errors due to misspelling and different acronyms used. We tried to mine all possible strings for a skill. For instance, we used the following phrases for MS SQL Server: (1) "microsoft sql server", (2) "ms sql", (3) "mssql", (4) "ms sql server", (5) "mssql server", (6) "sql-server", (7) "ms.sql". We obtained the frequencies of each skill as the result of fifth step.

In the sixth step, we formed document vectors for each job ad. These vectors only included the skills we obtained in the fifth step. We assumed that each job ad is represented with these skills. This type of approach is named as "bag of words" representation. We provided these document – word vectors to the seventh step as an input to build a topic model. We formed many topic models using Latent Dirichlet Allocation (LDA) [9] in the seventh step and finally came up with a reasonable model. This topic model consists of five topics. α and β parameters used to build this model was 0.85 and 0.76 respectively (for the details of these parameters refer to [10]). Using the topic model obtained, we clustered all job ads into five groups.

At the eighth step, we analyzed all of the findings and reported as presented in the next section.

We used Jupyter Notebook as a development environment. We developed our software using Python programming language. We used pandas library for data processing, nltk library for natural language processing and gensim library for building topic models. We also used Microsoft Excel for storing and analyzing data.

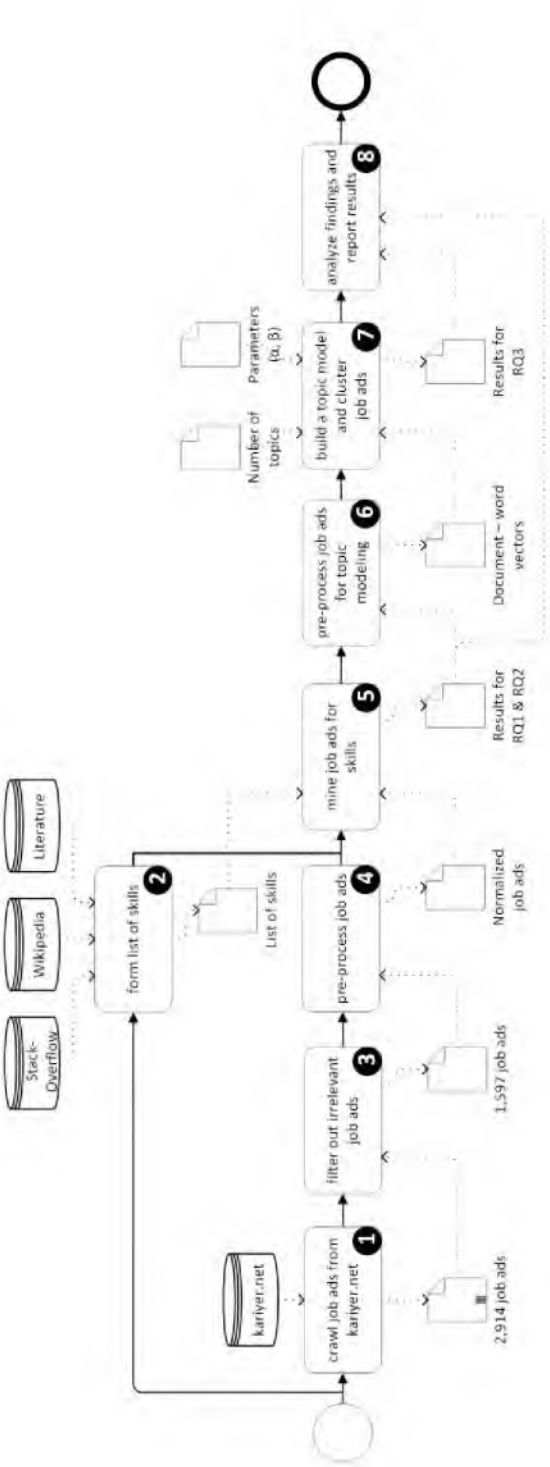


Fig. 1. The research method used in this study.

4 Results and Discussion

4.1 RQ1: Desired Technical Skills

Programming, Scripting, and Markup Languages. Fig. 2 (a) shows top 10 programming, scripting, and markup languages that are most frequently mentioned in computing job ads. Fig. 2 (b) displays top 10 programming, scripting, and markup languages that are used by developers according to [6]. SQL is by far the most desirable skill among languages and this finding is consistent with those reported in [7]. SQL is the third most used language according to [6]. JavaScript and Java are the two other languages desired by employers and used by developers as identified in this study, [7] and [6]. Python is the fastest-growing major programming language today [6]. [7] also identified Python as the fifth most desired language. On the other hand, Python is desired relatively less according to our analysis. [7] also reported that the highest gap between demand and supply will be in Python language skill within five years in Turkey. HTML/CSS and C# are the other two desired languages according to our analysis. HTML/CSS is the second most used language according to [6] and C# is reported as one of the most desired languages in top five by [7] and is ranked as seventh most used language by [6].

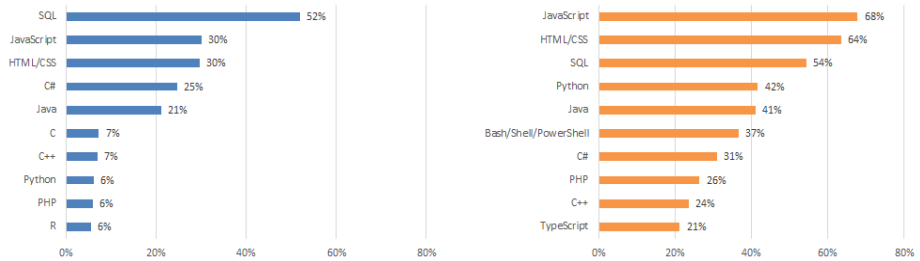
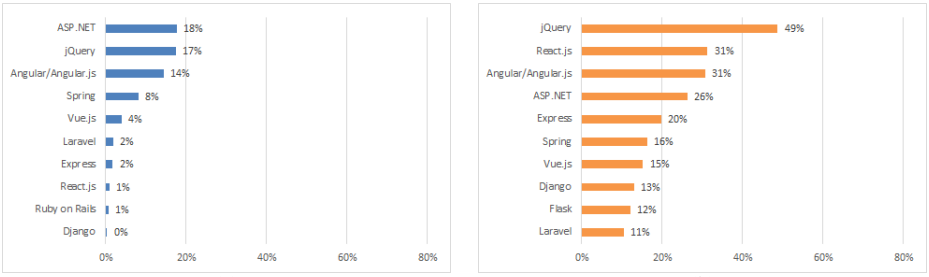


Fig. 2. Top 10 programming, scripting, and markup languages that are most frequently (a) mentioned in computing job ads; (b) used by developers according to [6].

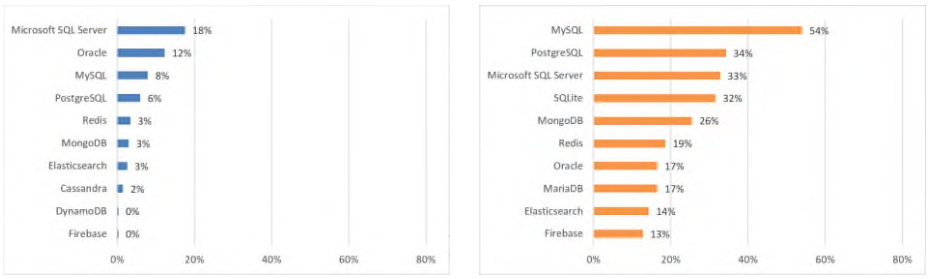
Web Frameworks. Fig. 3 (a) shows top 10 web frameworks that are most frequently mentioned in computing job ads. Fig. 3 (b) displays top 10 web frameworks that are used by developers according to [6]. ASP.NET is the most desired web framework in Turkish job ads. jQuery and Angular follow ASP.NET with a slight difference. jQuery is the most broadly used of these web frameworks according to [6]. In addition, [6] reported that this year more developers say they use React.js than Angular, a switch from last year. According to our dataset, employers in Turkish software industry do not seek for React.js skill that much. Spring and Vue.js are the other two web frameworks sought by employers and these two frameworks are in top ten most used web frameworks according to [6].



(a) (b)

Fig. 3. Top 10 web frameworks that are most frequently (a) mentioned in computing job ads; (b) used by developers according to [6].

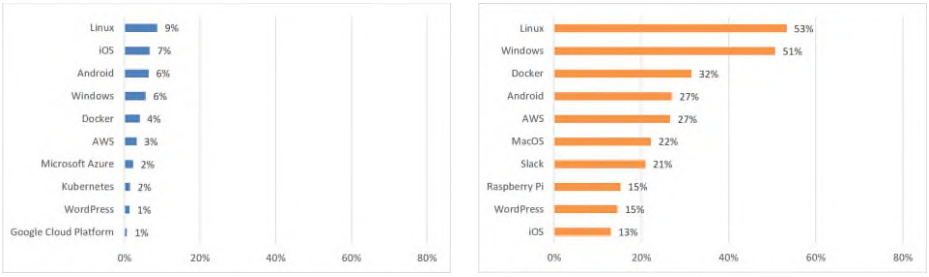
Databases. Fig. 4 (a) shows top 10 databases that are most frequently mentioned in computing job ads. Fig. 4 (b) displays top 10 databases that are used by developers according to [6]. Nearly one fifth of the employers look for professionals who can use MS SQL Server. Oracle, MySQL, and PostgreSQL are the other DBMSs whose knowhow is important to employers. According to [6] MySQL is the most commonly used database, like last year. PostgreSQL has taken the second spot this year, edging ahead of MS SQL Server, which is the most desired database based on our dataset. Another interesting remark is that SQLite, which is the fourth most commonly used database according to [6] is not included within top ten in our dataset.



(a) (b)

Fig. 4. Top 10 databases that are most frequently (a) mentioned in computing job ads; (b) used by developers according to [6].

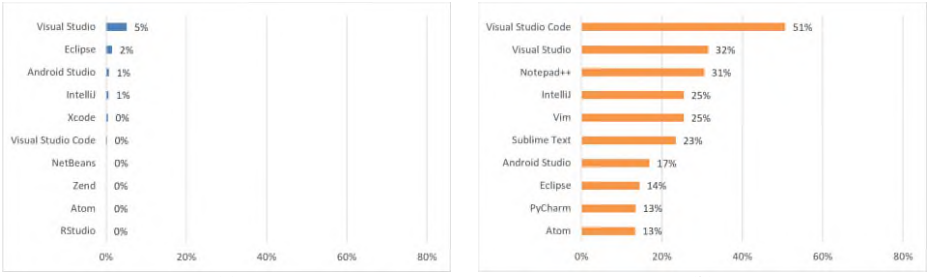
Platforms. Fig. 5 (a) shows top 10 platforms that are most frequently mentioned in computing job ads. Fig. 5 (b) displays top 10 platforms that are used by developers according to [6]. Linux is the most desired platform by Turkish employers and most common platform used worldwide according to [6]. Employers in Turkey look for developers working with mobile operating systems, i.e. iOS and Android. Windows operating system is still among the most used operating systems. Docker is also gaining attraction both worldwide and among Turkish employers.



(a) (b)

Fig. 5. Top 10 platforms that are most frequently (a) mentioned in computing job ads; (b) used by developers according to [6].

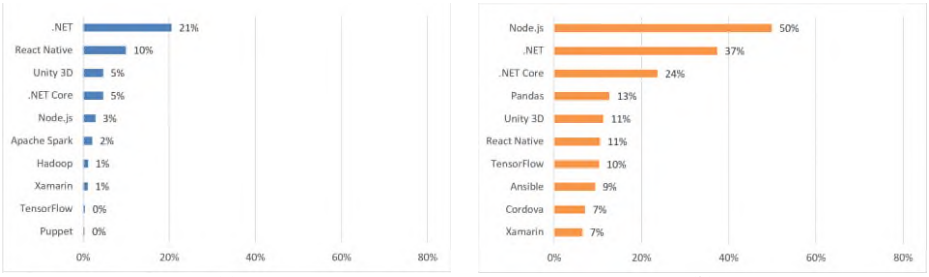
Development Environments. Fig. 6 (a) shows top 10 development environments that are most frequently mentioned in computing job ads. Fig. 6 (b) displays top 10 development environments that are used by developers according to [6]. The job ads in our dataset do not include too much information on development environments whose knowhow is important to employers. Visual Studio is the top development environment sought by Turkish employers and used worldwide [6]. On the other hand, Turkish employers do not ask for experience specifically on Visual Studio Code, which is the most commonly used development environment worldwide [6].



(a) (b)

Fig. 6. Top 10 development environments that are most frequently (a) mentioned in computing job ads; (b) used by developers according to [6].

Other Frameworks, Libraries, and Tools. Fig. 7 (a) shows top 10 other frameworks, libraries, and tools that are most frequently mentioned in computing job ads. Fig. 7 (b) displays top 10 other frameworks, libraries, and tools that are used by developers according to [6]. While Node.js is the most commonly used tool worldwide [6], employers in Turkey do not look for professionals with Node.js experience. More developers say they use .NET than .NET Core worldwide [6] and the same is valid for our dataset. React Native knowhow is looked by one tenth of employers. While Pandas is popular library worldwide [6], experience in Pandas is not important to Turkish employers.



(a) (b)

Fig. 7. Top 10 frameworks, libraries, and tools that are most frequently (a) mentioned in computing job ads; (b) used by developers according to [6].

Testing Tools. The survey whose results are reported in [6] does not include any specific question addressing software testing tools. Due to the increasing importance of software testing and availability of testing tools, we mined our dataset to explore the frequencies of testing tools in computing job ads. We formed a list of testing tools from Wikipedia under "software testing tools category" (https://en.wikipedia.org/wiki/Category:Software_testing_tools). We included the tools under "free software testing tools", "graphical user interface testing tools", "load testing tools", "security testing tools" and "unit testing frameworks". Our final list included 116 tools. Surprisingly only three tools, i.e. Selenium, NUnit, and JUnit, have been mentioned 19, 16, and 16 times respectively in 1597 job ads. A few of the tools have been mentioned less than 10 times in computing job ads. According to our dataset, employers do not seek for experience on software testing tools.

4.2 RQ2: Desired Soft Skills

Besides technical skills, soft skills (non-technical skills) also play an important role in software development. Therefore, employers generally prefer to include the soft skills they require for their open positions. To find out the frequencies of soft skills in our dataset, we formed a list of soft skills by exploring the earlier studies [3], [4], [8].

Fig. 8 shows the frequencies of soft skills in our dataset. There is a high demand for team work, communication and analytical/problem solving skills. Since software development is generally conducted by a team, the ability to work in teams is very important. Moreover, communication skills are also important and becoming more and more important since agile software development methods suggest frequent and oral communication in teams.

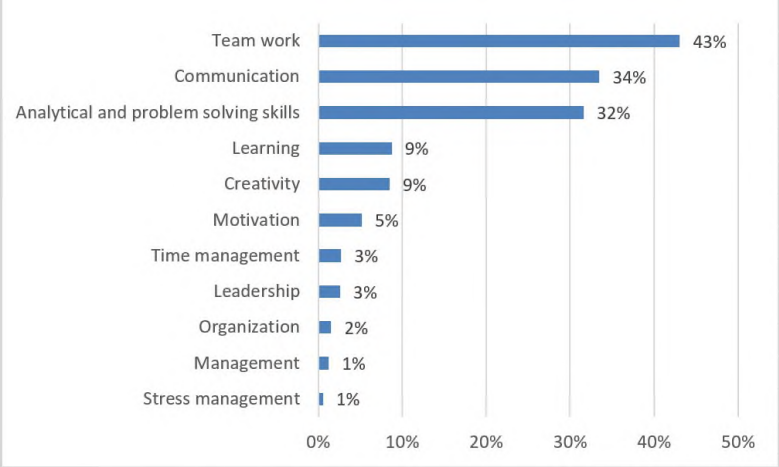


Fig. 8. Soft skills that are most frequently mentioned in computing job ads.

4.3 RQ3: Clustering Job Ads

Positions in software development can be classified according to skill requirements. Different positions require different sets of skills. Therefore, we clustered the job ads into five groups using topic modeling, similar to the approach in [5]. In topic modeling, number of topics is an input and does not have an initial value [10]. After several iterations, we obtained five topics to cluster the job ads in our dataset. Table 2 shows these five clusters and the most frequently mentioned top five technical/soft skills in these clusters.

The job ads in the first cluster mainly expect experience in languages and tools launched by Microsoft. This cluster includes the highest number of job ads, i.e. 429, so, according to our dataset, we can infer that employers look for experience on Microsoft products at most. The second cluster includes SQL, Java, JavaScript languages; Spring application framework for Java and Oracle database. The third cluster includes C and C++ languages and three operating systems, i.e. iOS, Android and Linux. As we can see from the frequencies, the third cluster does not have dominant skills. In the fourth cluster, only SQL has a relatively high frequency and the rest of the items in top five have less than 10% frequencies. On the other hand, the soft skills are much more frequently mentioned in this cluster compared to the others. Software developer, as most of the other professions, should use their soft skills more as they gain more managerial responsibilities [11], [12]. Therefore, we can infer that this cluster mostly includes job ads that are seeking for managerial roles. The dominant theme for the fifth cluster is web application development. The top five languages and tools sought by employers in this cluster are HTML/CSS, JavaScript, jQuery, SQL and Angular.

Table 2. Five clusters of job ads and the most frequently mentioned technical/soft skills in these clusters.

Cluster	# of job ads	Technical Skills		Soft Skills	
		Skill	Frequency (%)	Skill	Frequency (%)
1	429	SQL	74	Team work	36
		C#	67	Problem solving	28
		.NET	57	Communication	25
		ASP.NET	50	Learning	6
		MS SQL Server	47	Creativity	6
2	258	SQL	77	Team work	30
		Java	69	Communication	24
		Spring	41	Problem solving	19
		Oracle	40	Motivation	14
		JavaScript	29	Learning	9
3	270	C	32	Communication	36
		C++	29	Team work	30
		iOS	23	Creativity	17
		Android	22	Problem solving	13
		Linux	21	Learning	6
4	352	SQL	36	Team work	73
		JavaScript	7	Problem solving	65
		HTML/CSS	7	Communication	59
		Oracle	7	Learning	14
		Java	6	Creativity	10
5	288	HTML/CSS	84	Team work	41
		JavaScript	78	Problem solving	24
		jQuery	54	Communication	21
		SQL	49	Learning	8
		Angular	45	Creativity	7

5 Threats to Validity

Our study has some threads to validity. First, we crawled only kariyer.net as the data source. There are other web sites and environments for job search, hence kariyer.net does not represent the whole software industry in Turkey. On the other hand, it is one of the most popular and widely-used online platforms for job search. Moreover, we crawled the web site during two days in May 2019. Therefore our dataset reflects only a snapshot for skill requirements in Turkey for software development. Consequently, the results of our analysis have limited generalizability. To check the generalizability

of our results, we compared our results with another survey in Turkey [7] and a survey done worldwide by StackOverflow web site [6] whenever comparable data are available in the results of these surveys.Second, we manually filtered out irrelevant job ads. Therefore, this step is subject to a researcher bias. We think that this would be limited since most of the eliminated job ads are clearly irrelevant to software development, such as job ads seeking for networking experts, help desk technicians etc.

6 Conclusions

In this paper, we provide an overview of the technical and soft skills desired by employers in Turkish software industry. Our analysis reveals a substantial demand for experience in SQL, JavaScript and HTML/CSS languages in line with the recent trends worldwide. ASP.NET and MS SQL Server are the dominant web framework and database whose knowledge is needed most in job ads according to our dataset. Knowledge on Linux operating system is the most desired consistent with StackOverflow developer survey 2019 results. Visual Studio is the most desired development environment and .NET is the dominant framework. According to our dataset, experience on only a few software testing tools are sought by employers. The most desired soft skills are team work, communication and analytical/problem solving skills.

References

1. Litecky, C., Aken, A., Ahmad, A., & Nelson, J. (2009). Mining for computing jobs. IEEE software, 27(1), 78-85.
2. Hiranrat, C., & Harncharnchai, A. (2018, July). Using Text Mining to Discover Skills Demanded in Software Development Jobs in Thailand. In Proceedings of the 2nd International Conference on Education and Multimedia Technology (pp. 112-116). ACM.
3. Ahmed, F., Capretz, L. F., & Campbell, P. (2012). Evaluating the demand for soft skills in software development. It Professional, 14(1), 44-49.
4. Papoutsoglou, M., Mittas, N., & Angelis, L. (2017, August). Mining People Analytics from StackOverflow Job Advertisements. In 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA) (pp. 108-115). IEEE.
5. Föll, P., Hauser, M., & Thiesse, F. (2018). Identifying the Skills Expected of IS Graduates by Industry: A Text Mining Approach.
6. StackOverflow Developer Survey Results (2019). https://insights.stackoverflow.com/survey/2019. Last accessed: 23 June 2019.
7. Taşıtman, A. (2019). Türkiye Teknoloji Sektörü Durum Analizi Araştırma Raporu.
8. Ahmed, F., Fernando Capretz, L., Bouktif, S., & Campbell, P. (2012). Soft skills requirements in software development jobs: a cross-cultural empirical study. Journal of systems and information technology, 14(1), 58-81.
9. Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. Journal of machine Learning research, 3(Jan), 993-1022.
10. Blei, D.M. (2012). Probabilistic topic models. Commun. ACM 55, 4 (April 2012), 77-84. doi: https://doi.org/10.1145/2133806.2133826
11. Katz, R. (1955). Skills of an effective administrator. Harvard Business Review, Jan-Feb.
12. Northouse, P. G. (2018). Leadership: Theory and practice. Sage publications.

Kaynak Kodlardaki Kötü Kokuların Analizinde Hata Komşuluk Etkisi

Rahime Belen Sağlam[1], Özkan Kılıç[2], and Yusuf Şevki Günaydın[3]

[1] Ankara Yıldırım Beyazıt Üniversitesi,Ankara, Türkiye rbsaglam@ybu.edu.tr
[2] Ankara Yıldırım Beyazıt Üniversitesi,Ankara, Türkiye ozkankilic@gmail.com
[3] Ankara Yıldırım Beyazıt Üniversitesi,Ankara, Türkiye ysgunaydin@ybu.edu.tr

Özet. Kaynak kodlardaki kötü kokular kod kalitesini idame edebilmek için çözümlenmesi gereken tasarım sorunlarıdır. Literatürde kodlardaki kötü kokuların tespiti için önerilmiş bir çok araç ve teknik bulunmaktadır, ancak, bu teknikler aksiyon alınmasını gerektirmeyen yalancı pozitif sonuç üretme eğilimindedir. Dolayısıyla, kötü kokular arasından aksiyon alınmasını gerektirenleri tespit etmeye yönelik ileri yöntemler geliştirilmiştir. Bu yaklaşımlar önceliğine bakmaksızın son revizyondan önce çözümlenmiş tüm uyarıların aksiyon alınması gereken uyarılar olduğu varsayımına dayanır. Bu çalışmada, çözümlenmiş uyarılar üzerine bir analiz gerçekleştirdik ve geliştiricilerin çözümlemek üzere uyarı seçerken kullandıkları faktörleri açığa çıkarmaya çalıştık. Çözümlenen düşük öncelikli uyarılara odaklanarak, bu uyarılarla aynı sınıfta bulunan diğer uyarıları inceleyip geliştiricilerin düşük öncelikli uyarıları yüksek öncelikli uyarılar ile aynı sınıfta bulunduğu durumda çözme eğiliminde olduğunu gözlemledik. Bu bulgularımız literatürdeki fazla basite indirgenmiş varsayımı değiştirerek aksiyon alınması gereken uyarı tespiti araştırmalarındaki doğruluğu arttırabilecek niteliktedir.

Anahtar Kelimeler: Yazılım Kalitesi, Makine öğrenmesi, Kusurlu kod, Aksiyon Alınabilir Uyarı, Sınıflandırma.

Bug Neighbourhood Effect on Analysis of Code Smells in Source Codes

Rahime Belen Sağlam[1], Özkan Kılıç[2], and Yusuf Şevki Günaydın[3]

[1] Ankara Yıldırım Beyazıt University,Ankara, Türkiye rbsaglam@ybu.edu.tr
[2] Ankara Yıldırım Beyazıt University,Ankara, Türkiye okilic@ybu.edu.tr
[3] Ankara Yıldırım Beyazıt University,Ankara, Türkiye ysgunaydin@ybu.edu.tr

Abstract. Code smells are the indicators of design flaws or problems in source code which require to be resolved to maintain code quality.

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

In the literature, various tools and techniques have been proposed to detect code smells; however, those techniques tend to generate spurious false positive warnings, which do not require an action to be taken. Consequently, further approaches have been developed for identification of actionable alerts among them. Those approaches rely on a naive assumption and each alert that is resolved before the latest revision is assumed to be actionable disregarding the priority of it. In this study, we have performed some analysis on alerts that were resolved by the developers and tried to uncover the factors that professional developers use to select code smells to resolve. Focusing on the low priority alerts that are resolved, we investigated the other smells located in the same classes (i.e., collocated smells) and observed the tendency of developers to resolve the low priority issues when they co-occur with the high priority ones in the same file. In addition, applying classification algorithms using the features that cover the information about the neighborhoods of an alert, we observed that the accuracy of actionable alert detection significantly increases.

Keywords: Software Quality, Machine Learning, Code Smells, Actionable Alerts, Classification

1 Introduction

Quality of program source code has been one of the key concerns in software engineering and code smells [8], symptoms of decays in code quality, received great attention both from the academic researchers and industry. A possible way to improve the quality of source code and to reduce bugs are code reviews, in which other developers review changes. However, this approach relies on human inspection and requires considerable human effort. In this context, static code analysis tools introduced an excellent opportunity for developers by analysing the code in an automated way and providing unbiased, objective results.

Static code analysis tools perform analysis of source code with the aim of highlighting potential issues that can arise in a software system. Automatically detected issues help developers to maintain high quality software by detecting issues such as uninitialised or unused variables, empty catch blocks, poorly commented and organized code (too long lines or methods) etc. Over the years, different tools have been proposed with different facilities including automatically checking code styles, detecting bugs and prioritizing them. Nowadays, studies on static code analysis have matured enough that industrial grade tools are used in commonly in both open source and industrial projects.

Despite their advantages, the main challenge complicating the adoption of these tools is spurious false positive warnings (i.e., alerts that are not actual issues that require an action). Taking the program as an input, a static analysis tool outputs lists of alerts where an alert is an indication that a check the tool performs is violated. As the alert is reported to the developers, it may trigger an action and the developers may choose to react to it or they can choose to ignore the alert. In this context, the number of generated alerts is important for

a developer to make the right decision for each alert. However, static analysis tools often over-estimate possible program behaviors and generate alerts that do not correspond to true defects [9]. Kremenek et al. [13] report that at least 30% of the warnings reported by sophisticated tools are false positives. In order to improve the output of the static analysis tools, several studies have been conducted to detect 'actionable alerts' or true positives that need to be resolved by developers. In those studies, researchers proposed several post-processing methods by classifying or ranking the list of generated alerts[1] [2] [4] [5] [14][15]. In those studies, an actionable alert is defined as the one that will be acted upon.

Because getting the actionable alerts by developer feedback is expensive, Heckman et al. proposed a naive approach to classify the alerts: if an alert is still open in the latest revision it is assumed as un-actionable whereas all others, that disappear before the latest revision, are assumed to be actionable, unless the file containing the alert disappeared [10]. Those alerts that belong to a removed file are discarded.This definition has been commonly accepted and used in several studies. Allier et. al slightly changed the definition of un-actionable alerts. Arguing that all alerts that are not resolved in the latest revision cannot be defined as un-actionable, they defined them as discarded [2]. However, they kept the definition of actionable alerts the same.

Disregarding the definition of non-actionable alert definition, in this study we focused on actionable alert definition and analysed the risk of defining alerts that are taken action before the latest revision as actionable. We argue that the developer's decision to take an action about an alert may differ considering the other alerts generated for the same file and the action taken by the developer does not necessarily mean that the alert is important enough to be resolved. Most of the static code analysis tools provide some details about the alerts including type and priority. It is a known fact that engineers generally focus on higher-priority alerts because finding important bugs is the primary goal [14]. They tend to ignore low priority alerts. This practice is inline with the assumption made in actionable alert definition. However, the definition leads all the resolved alerts to be labelled as actionable disregarding the priority. We argue that the low priority alerts resolved by the developers may cause misleading results in the studies based on this definition. It also decreases the possibility of low priority issues to be detected as non-actionable and provides a challenge for decreasing false positives. Consequently, in this study, we focused on resolved low priority issues and analyzed possible reasons behind the action taken by the developer. We argue that the developers tend to resolve low priority issues when they co-exist with the high priority ones in the same file while they can ignore them when they appear on their own.

In order to test this hypothesis, we collected alerts from the different versions of two open source platforms, namely FIWARE and JDOM, and utilized some features for each alert. We have designed two experiments where in one of the experiments we have expanded the feature set with the proposed feature related to neighborhood information. We ran different machine learning based classification algorithms to detect actionable alerts and observed that the success

rate of actionable alert identification increases significantly when neighborhood information is used as a new feature.

2 Data set

We used PMD static code analysis software in order to collect alerts from two open source platforms, FIWARE and JDOM. PMD allows to run a set of static code analysis rules on source code files and generates a list of problems that conflict with those rules [3].

FIWARE is an open source framework which provides API modules, called Generic Enablers, to rapidly develop smart solutions. Generic Enablers are set of building blocks which make it easy to develop smart Internet Applications. Within this study, we have focused on one of the GE, AuthZForce, which has 12 versions between the years of 2016 and 2018. This GE consists of Java codes [7]. Similarly, JDOM is an open-source Java-based document object model for XML and it has 8 versions released between the years of 2011 and 2013 [11].

In order to obtain alerts that were resolved or ignored by the developers, first, we have run static code analysis on the each version of the projects. As we have alerts generated by PMD for each version of the projects, we have defined an approach to understand if a specific alert generated for a specific version has been resolved in letter versions. For this purpose, we focused on each consecutive releases and compared all the alerts in the same functions between the consecutive versions. If an alert with the same priority and the same rule ID is not detected in the letter versions of a method, we assumed that the alert has been resolved, which indicated that it was actionable. The remaining errors were assumed to be non-actionable. The following 9 features were utilized for each alert.

· **Priority** It is an integer value produced by the static code analysis software between 1 and 5 where 5 indicates the highest priority.
· **Line** It is an integer value indicating the line number of the alert within a file.
· **FileLOC** The total lines of code of the file with the alert.
· **Line/FileLOC** Line number of the error in the file divided by the total lines of code in that file
· **RuleSet** It is a string value stating the category of the error type. PMD has 8 categories including Code Style, Error Prone, Security, Performance etc.
· **Rule** It is a string value stating the type of the error. There are more than 100 error types in PMD.
· **VersionNumber** It is an integer value stating the version of the platform in which an alert arises. We collected data from 12 versions of FIWARE and 8 versions of JDOM.
· **ErrBtw2Version** It is an integer number stating the number of errors between two consecutive versions.
· **DayBtw2Version** It is an integer number stating the number of days between the release dates of two consecutive versions.

We have run classification algorithms using these features. Then, we applied the same algorithms on our new feature set that has been expanded by 10 more features that reflect the neighborhood between alerts:

· **SP[1/2/3/4/5]** Five integer valued features indicating the number of resolved alerts with priority values 1 to 5 within the same file.
· **UP[1/2/3/4/5]** Five integer valued features indicating the number of unresolved alerts with priority values 1 to 5 within the same file.

We have worked on 22,933 alerts generated by PMD for FIWARE 1,029 of which were resolved and 21,904 were not. Similarly, a total number of 186,071 alerts were detected for JDOM. 4,825 of them were resolved by the developers while 181,246 were ignored.

3 Methodology and Results

In order to test our hypothesis, we focused on low priority alerts that were resolved by the developers and computed the number of higher priority alerts in the same file. The results for JDOM and FIWARE data can be seen in Table 1 and Table 2 respectively. The tables show that most of the low priority resolved alerts appear in the same file with higher priority alerts. This observation constitutes the basis of the current study and has motivated us to apply classification algorithms on features including neighbourhood information.

Table 1. Bug Neighbourhood on JDOM Data

	Actionable Alerts			Nonactionable Alerts		
	Priority 1	Priority 2	Priority 3	Priority 1	Priority 2	Priority 3
Actionable Priority 4	31	1	236	6	32	104
Actionable Priority 5	208	38	2889	138	120	2343

Table 2. Bug Neighbourhood on FIWARE Data

	Actionable Alerts			Nonactionable Alerts		
	Priority 1	Priority 2	Priority 3	Priority 1	Priority 2	Priority 3
Actionable Priority 5	0	5	718	39	34	817

We ran k-Nearest Neighborhood (kNN), Support Vector Machines(SVM), and Artificial Neural Networks (ANN) algorithms on the four data file as explained in the previous section. For kNN, we empirically selected 5 as k value

and Euclidean distance for classification. In order to train an SVM, a Gaussian kernel was employed with the aim of minimizing both the estimation and approximation errors. Finally, the ANN had 3 hidden layers with ReLU activation function and an output neuron with sigmoid function. The network was fully connected.

For all experiments, 20% of data were split for testing while 80% of them were used for training the classifiers. The following tables illustrate precision, recall and f-1 score for the experiments.

Table 3. Experiment Results on FIWARE Data

	Without neighborhood info			With neighborhood info		
	KNN	SVM	ANN	KNN	SVM	ANN
precision	77.51%	49.76%	63.64%	86.60%	72.25%	89.00%
recall	85.26%	95.41%	85.81%	89.60%	90.96%	86.11%
f-1 score	81.20%	65.41%	73.08%	88.08%	80.53%	87.53%

Table 4. Experiment Results on JDOM Data

	Without neighborhood info			With neighborhood info		
	KNN	SVM	ANN	KNN	SVM	ANN
precision	74.16%	61.85%	65.82%	80.98%	64.70%	79,65%
recall	84.96%	87.10%	70.25%	90.56%	93.12%	82,95%
f-1 score	79.20%	72.34%	67.96%	85.50%	76.35%	81,27%

The results indicate that when bug neighborhood information is added as additional features, all the machine learning algorithms are found to produce higher precision, recall and f1-score. K-nearest neighbors algorithm outperformed the others in both experiments.

4 Conclusion

In this study, we aimed to investigate the effect of bug neighborhood on developers while selecting alerts to resolve. For this purpose, the alerts between different versions of two open source platforms, FIWARE and JDOM, have been collected. The distribution of the resolved low priority alerts have been investigated and it has been observed that the majority of them co-exist with high priority alerts in the same file. In order to enrich our findings, we have proposed some features

that can reflect the neighborhood information between alerts with different priorities and have run experiments to classify alerts. It has been observed that classifiers can identify actionable alerts better when the neighborhood information is covered in the features which can be due to the success on detecting low priority issues that are resolved by the developers.

Machine learning methods have impressive success in predicting human decisions when trained on large amount of data [6] [12]. Software developers are no exceptions.

When a developer decides to take an action against an error, he or she generally goes with the one with the high priority. The developer usually searches and finds the alert, and then implements the change plan to eliminate the alert. These are the major cost of steps taken against an alert. This study indicates that software developers tend to solve low priority alerts in the vicinity of high priority ones which can be explained with the low effort required to fix it. As the developer accesses to a file with high priority alerts, she can resolve the low priority ones with a relatively limited effort compared to the one whose file is not accessed and edited. Therefore, she could perceive that solving the low priority one in the same context would not take much effort. Finally, the authors would like to recommend that static-code analysis tools include the concept of bug neighborhood in prioritizing the alert.

References

1. A. Aggarwal and P. Jalote. Integrating static and dynamic analysis for detecting vulnerabilities. In *30th Annual International Computer Software and Applications Conference (COMPSAC'06)*, volume 1, pages 343–350. IEEE, 2006.
2. S. Allier, N. Anquetil, A. Hora, and S. Ducasse. A framework to compare alert ranking algorithms. In *2012 19th Working Conference on Reverse Engineering*, pages 277–285. IEEE, 2012.
3. PMD Source Code Analyser. Pmd:an extensible cross-language static code analyzer.pmd source code analyzer, 2019. Accessed = 2019-06-23.
4. C. Boogerd and L. Moonen. Prioritizing software inspection results using static profiling. In *2006 Sixth IEEE International Workshop on Source Code Analysis and Manipulation*, pages 149–160. IEEE, 2006.
5. C. Csallner and Y. Smaragdakis. Check'n'crash: combining static checking and testing. In *Proceedings of the 27th international conference on Software engineering*, pages 422–431. ACM, 2005.
6. D. Silver et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(484), 2016.
7. FIWARE FOUNDATION. Fiware: The open source platform for our smart digital future, 2019. Accessed = 2019-06-23.
8. M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts. *Refactoring: improving the design of existing code.* Addison-Wesley Professional, 1999.
9. S. Heckman and L. Williams. A model building process for identifying actionable static analysis alerts. In *2009 International Conference on Software Testing Verification and Validation*, pages 161–170. IEEE, 2009.

10. S. Heckman and L. Williams. A comparative evaluation of static analysis actionable alert identification techniques. In *Proceedings of the 9th International Conference on Predictive Models in Software Engineering*, page 4. ACM, 2013.
11. JDOM. Jdom, 2019. Accessed = 2019-06-23.
12. J. Kleinberg, H. Lakkaraju, J. Leskovec, J. Ludwig, and S. Mullainathan. Human decisions and machine predictions. *Q. J. Econ.*, 133:237—-293, 2017.
13. T. Kremenek and D. Engler. Z-ranking: Using statistical analysis to counter the impact of static analysis approximations. In *International Static Analysis Symposium*, pages 295–315. Springer, 2003.
14. J. R. Ruthruff, J. Penix, J. D. Morgenthaler, S. Elbaum, and G. Rothermel. Predicting accurate and actionable static analysis warnings: an experimental approach. In *Proceedings of the 30th international conference on Software engineering*, pages 341–350. ACM, 2008.
15. H. Shen, J. Fang, and J. Zhao. Efindbugs: Effective error ranking for findbugs. In *2011 Fourth IEEE International Conference on Software Testing, Verification and Validation*, pages 299–308. IEEE, 2011.

Mobil Uygulamalarda Gereksinim Belirlemede Başarıyı Arttıracak Bir Yöntem: Kitap Paylaşım Uygulaması Örneği

Birsen Gülden Özdemir[1] and Mustafa Zahid Gürbüz[2]

[1] Doğuş Üniversitesi, İstanbul, TR, bgozdemir@dogus.edu.tr
[2] Doğuş Üniversitesi, İstanbul, TR, zgurbuz@dogus.edu.tr

Özet. Akıllı mobil cihazlarının yaygın bir şekilde kullanımı bu cihazlarda çalışacak uygulamaları ön plana çıkarmaktadır. Kullanıcılar istedikleri yerde ve anda erişebiliyor olmalarından dolayı mobil uygulamalara yönelmekte, gün geçtikçe daha fazla tercih etmektedirler. Mobil uygulamalarda gereksinimlerin net bir şekilde belirlenmesi projeleri başarıya götüren en önemli adım olarak karşımıza çıkmaktadır. Belirlediğimiz gereksinim belirleme yöntemini bu çalışma ile paylaşmadaki amacımız mobil uygulamalarda başarıyı arttırmayı hedefleyen uygulama geliştiricilerine yol göstermektir.

Anahtar Kelimeler: Mobil uygulamalar; kullanıcı gereksinimleri.

A Method to Increase Success in Determining Mobile Application Requirements: Book Sharing Application Example

Birsen Gülden Özdemir[1] and Mustafa Zahid Gürbüz[2]

[1] Dogus University, Istanbul, TR, bgozdemir@dogus.edu.tr
[2] Dogus University, Istanbul, TR, zgurbuz@dogus.edu.tr

Abstract. The widely use of smart mobile devices bring into prominence the applications that will work on these devices. Since users can access anywhere and at any time, they prefer to use mobile applications more. The clear determination of the requirements in mobile applications is the most important step leading to successful projects. The purpose of this study is to guide the application developers aiming to increase the success in mobile applications.

Keywords: Mobile applications; user requirements.

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

1 Giriş

Yazılım projeleri, takvim ve bütçe kısıtlamaları altında müşteri gereksinimlerine ve teknolojik olanaklara uygun olan kalite, kullanıcı dostu uygulamalar geliştirmeyi amaçlamaktadır. Akıllı mobil cihazlarının yaygın bir şekilde kullanımı bu cihazlarda çalışacak uygulamaları ön plana çıkarmaktadır. Kullanıcılar istedikleri yerde ve anda erişebiliyor olmalarından dolayı mobil uygulamalara yönelmekte, gün geçtikçe daha fazla tercih etmektedirler.

Literatürde bir projenin başarısını ölçebilmek için Kritik Başarı Faktörleri (KBF) (Critical Success Factors) tanımı önerilmiştir [1]. Bu tanıma göre göre, 37 adet kriter belirlenmiş olup bu kriterler 4 sınıfa ayrılmıştır. Bunlar sırasıyla, organizasyon faktörler, takım faktörleri, müşteri faktörleri ve proje faktörler olarak belirtilmiştir.

Dinçer ve Garousi [2]'nin çalışmasında başarısız olan 100'ün üzerinde proje incelenmiş ve burada da görüldüğü gibi büyük çoğunluğu gereksinim belirlemedeki eksiklikler veya yanlışlıklardan kaynaklanmaktadır. Gereksinimlerin değiştirilmesi, proje ilerledikçe ve sonraki safhalara geçtikçe yeni gereksinimlerin adaptasyonunun zorlaşmasından dolayı, henüz gereksinim aşamasında iken Prototip kullanmak olumlu sonuçlar ortaya çıkarmaktadır. Gereksinim analizi tam yapılmadan ortaya çıkartılan projeler, işlerin önceliklendirilmesi veya gereksiz işlerin çıkartılması gibi aşamalar eksik olduğu için başarısız sonuçlar ortaya çıkmaktadır[2].

Askaroğlu ve Yılmaz'ın çalışmasında belirtildiği gibi, projelerde, gereksinimlerin ekip toplantılarında gözden geçirilmesi, kullanılacak teknolojilerin neler olması gerektiğine karar verilmesi ve bunların kayıt altına alınıp izlenmesi büyük önem arz etmektedir [3].

Albayrak [9] yaptığı çalışmada prototip kullanmayı seçen veya kodlamayı seçen öğrenciler arasında gereksinim belirleme başarısını araştırmıştır. Elde ettiği sonuçlara göre prototip kullanmayı seçen öğrencilerin gereksinim belirlemede daha verimli oldukları gözlemlenmiştir.

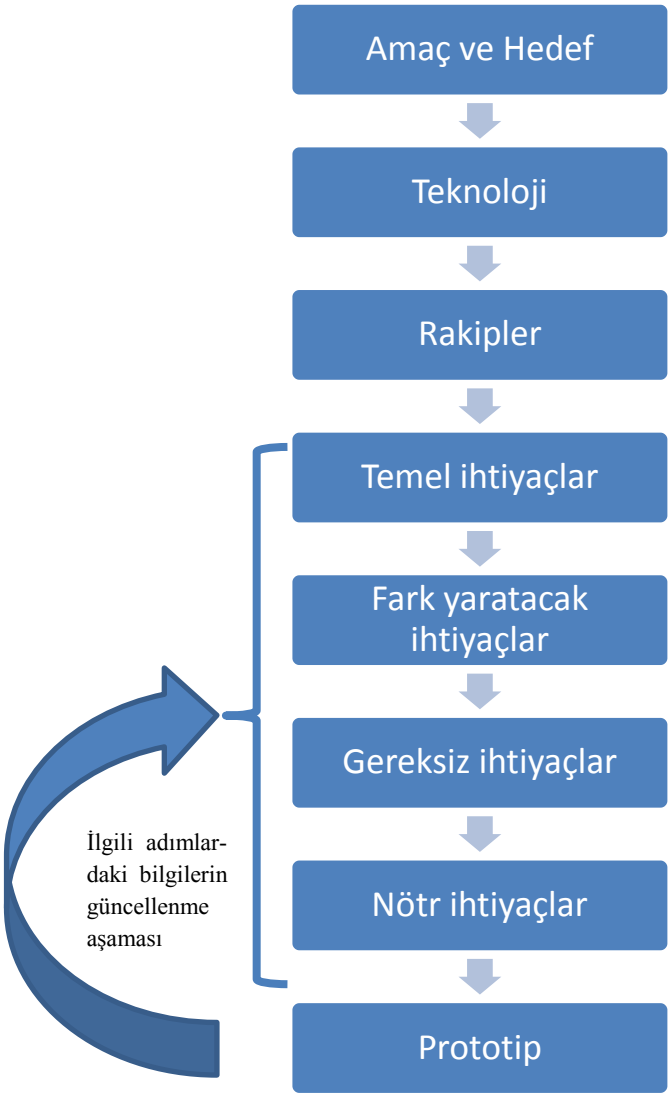
Nezcan vd [10]'un çalışmasında değindiği üzere, uygulamalarda yenilik süreci, gereksinimin belirlenmesi ve yeni fikirlerin üretilmesi ile başlamaktadır. Öncelikle hedef kullanıcıların talepleri göz önünde bulundurulmalı, sonra yeni fikirler üretilmelidir. Bu sürece etkileşimli yenilik süreci denilmektedir.

Köklü [11]'in de bahsettiği gibi projedeki tüm paydaşlarla doğru ve etkin bir iletişimin sağlanması temel ihtiyaçların ortaya çıkmasında etkili bir rol oynamaktadır. Buna paralel olarak Giray [12] araştırmasında paydaşlarla görüşmeler sonucu gereksinimleri belirleme yoluna gitmiştir.

Mobil uygulamalarda gereksinimlerin net bir şekilde belirlenmesi projeleri başarıya götüren en önemli adım olarak karşımıza çıkmaktadır. Belirlediğimiz gereksinim belirleme yöntemini bu çalışma ile paylaşmadaki amacımız mobil uygulamalarda başarıyı arttırmayı hedefleyen uygulama geliştiricilerine yol göstermektir.

2 Gereksinim Belirlemede Başarıyı Arttıracak Bir Yöntem

Yapılan araştırma ve incelemelerin ardından gereksinim belirlemede ilk adım, asıl amaç ve hedef kullanıcı kitlesinin belirlenmesi olması gerektiği sonucuna varılmıştır. Şekil 1'de yöntemin ilk adımı ve ardından gelen adımlar genel hatlarıyla belirtilmiştir.



Şekil 1. Gereksinim belirlemede başarıyı arttıracak bir yöntem

2.1 Asıl Amaç ve Hedef Kullanıcı Kitlesinin Belirlenmesi

Hedef kullanıcı kitlesini belirlemek mobil uygulamanın kapsamını belirlemede ilk atılması gereken adım olmalıdır.

2.2 Kullanılacak Teknolojilerin Belirlenmesi

Kullanılacak mobil platformların belirlenmesi ve uygulamanın çevrimiçi ya da çevrim dışı olup olmadığının belirlenmesi gerekmektedir.

2.3 Benzer Uygulamaların İncelenmesi

Piyasada mevcut olan uygulamaları incelemek gereksinim belirlemede önemlidir. Bu çalışmada özellikle bu konuda inceleme yapılmıştır. Bu incelemeler üç yönden yapılması önerilmiştir.

Barındırdığı Özelliklerin Tespit Edilmesi

Mevcut uygulamaların içinde bulundurduğu özelliklerin belirlenmesi aşamasının tamamlanması gerekmektedir. Aynı piyasada hizmet eden uygulama yapılacaksa o zaman rakiplerden eksik temel özellikler bulunması uygulamanın rekabet gücünü düşürecektir.

Olumlu Yorumlara Sahip Özelliklerin Tespit Edilmesi

Olumlu yorumlar genellikle temel özelliklerin belirlenmesinde önemli rol oynar. Gereksinimlerin önceliklendirilmesi aşamasında burada görülen özellikler dikkate alınmalıdır.

Olumsuz Yorumların Değerlendirilmesi

Olumsuz yorumlar genellikle fark yaratacak özelliklerin tespit edilmesinde faydalı olacaktır. Rakiplerin sağlayamadığı kullanıcı talepleri gereksinimler içerisine almak uygulamayı ön plana çıkarmada etkili olacaktır.

2.4 Kullanıcıların Temel İhtiyaçlarının Belirlenmesi

Öncelikle tüm ihtiyaçların listesi çıkartılmalıdır. Burada asıl amaca yönelik özelliklerin yanı sıra hedef kitleden alınmış özelliklere de yer verilmelidir. Aynı zamanda mevcut uygulamaların barındırdığı özelliklerle olumlu ve olumsuz yorumlardan elde edilecek özellikler listelenmelidir. Proje ekibinin tecrübesi ve beyin fırtınası seansı sonuçları da göz önüne alınarak olabildiğince fazla özellik listelenmelidir. Daha sonra bu özellikler içerisinden bazıları temel özellik, bazıları fark yaratacak özellik, bazıları da gereksiz özellik ve bazıları da nötr yapıda özellik olarak belirlenmelidir.

Akdur vd. [4]'nin de uyguladığı gibi, en son güncel bilgileri kapsaması için tespit edilen ihtiyaçların bir anket ile kullanıcıya sorulması, son kullanıcının sürece dahil edilmesini sağlayacaktır.

Asıl Amaca Uygun İhtiyaçlar

Olabilecek tüm özellikler listelendikten sonra bunların içerisinden asıl amaca uygun olmayanların elenmesi gerekmektedir. Bazen çok güzel gibi görünen bir özellik uygulamanın amacından sapmasına ve projeyi başarısızlığa sürükleyebilir [1]. Bu nedenle asıl amaca odaklanmak gerekir.

Mobil Platforma Uygunluğu

Uygulama bir mobil uygulama olarak düşünüldüğünde mobil platforma uygun olmayan gereksinimler temel ihtiyaçlara eklenmemelidir. Örneğin, çok sayıda veri girişi gerektiren ve fazla vakit alan özellikler temel gereksinimlerde bulunmamalıdır.

Kararsız Kalınan Özellikler

Temel özelliğe dâhil edip etmeme kararı veremediğimiz özelliklerin temel ihtiyaçlara eklenmemesi gerekmektedir. Bu özellikler fark yaratan yada nötr özellik olarak daha sonra tekrar değerlendirilebilir.

2.5 Fark Yaratacak İhtiyaçlarının Belirlenmesi

Bu özellikler aslında proje fikir aşamasında belirtilmiş olması gerekmektedir. Ancak müşteri analizi yapıldıktan sonra güncellenmelidir. Mevcut uygulamada karşılaşılan olumsuz yorumlardan elde edilen özellikler bu aşamada çok faydalı olacaktır. Aynı zamanda son kullanıcının güncel ihtiyaç, ilgi ve yönelimiyle örtüşen özellikleri kullanıcıyla birebir iletişime geçerek belirleyip uygulamanın gereksinimlerine eklemek gerekmektedir.

2.6 Gereksiz İhtiyaçlarının Belirlenmesi

Geliştirme süreci boyunca netleştirilmiş gereksinimler dışında kalan özelliklerle ilgili uygulama kapsamı dışında kalan çalışmalara zaman ve iş gücü ayrılmasını engellemek için düzenli (belirlenmiş zaman aralıklarında) kontrollerin yapılması gerekmektedir.

2.7 Nötr İhtiyaçlarının Belirlenmesi

Bazı özellikler kullanıcı açısından çok önemli olmamakla birlikte tamamen çıkartmak da istenmeyebilir. Bu durumda bu ihtiyaçların önceliklendirmesini düşük tutup yeterli kaynak olması durumunda uygulamaya eklenebilir.

2.8 Prototip Tasarımları Yardımıyla Kullanılabilirlik Testleri Yapılması

Son kullanıcılarla tasarlanan prototipler üzerinden düzenli olarak (belirlenmiş zaman aralıklarında) fonksiyonel yeterlilik ve kullanılabilirlik testleri yapılmalıdır. Bu adımda elde edilen verilerle 2.4.-2.7 arasındaki adımlardaki bilgiler güncellenmelidir.

3 Kitap Paylaşım Uygulaması

Yukarıda anlatılanlar ışığında örnek bir uygulama olarak kitap paylaşım uygulaması için gereksinimler belirlenmiştir. Arayüz tasarımı yapılırken kullanıcı dostu olması açısından mümkün olduğunca sade bir tasarım olmasına dikkat edilmiştir.

3.1 Asıl Amaç ve Hedef Kullanıcı Kitlesi

Asıl amaç insanların okuduğu ve beğendiği basılı kitapları diğer insanlarla paylaşması hedeflenmektedir. Bu doğrultuda hedef kitle her yaş grubu için kitap okumayı seven ve kitaba sahip olan kişilerdir.

3.2 Kullanılacak Teknolojilerin Belirlenmesi

Mobil uygulama piyasasında %99,5 pazar payına sahip[13] Android ve IOS işletim sistemleri ile uyumlu olarak kodlanması hedeflenmektedir.

3.3 Benzer Uygulamaların İncelenmesi

Horouz uygulaması bir mobil uygulama olarak çalışıyor. Takas etmek istenilen kitap uygulamada paylaşılıp çevredeki diğer kitap severler arasında kitap takası yapabilmektedir [5].

Horouz'a benzer şekilde çalışan Kipu uygulaması offline ve online kitap takası yapmaktadır. Kullanıcı deneyimi iyileştirmek için aynı zamanda ücretli olarak gönderim de sağlıyor [6].

Kitap dostum uygulaması, kitap takası yapmayı sağlamakla birlikte kitap okumayı sevenler için bir sosyal ağ oluşturmayı hedeflemektedir [7].

Kitapodam uygulaması, okuduğunuz kitabı başkalarının da okumasına yardımcı olmak için ücretsiz olarak takas edilebilen ve bu takasın başarısını da puan sistemi ile kullanıcılara geri bildirimde bulunan bir uygulamadır [8].

Barındırdığı Özelliklerin Tespit Edilmesi

İncelenen uygulamaların içerisinde yer alan özellikler aşağıda toplu olarak belirtilmiştir:

- Kitap takası için kullanıcı elindeki kitabın fotoğrafını çekerek sisteme yüklüyor.
- Kitap ile birlikte kullanıcının konumu da sisteme kaydediliyor.

- Kitap arayan kişiye harita üzerinden kitabın yeri gösteriliyor.
- Kitap takas etmek isteyen kişiler mesajlaşarak kendi aralarında takası gerçekleştiriyor.
- Kitap listesi sistemde yüklüdür. Kullanıcılar kitapları listeden seçiyor.
- Okuduklarım, okumak istediklerim, kütüphanem şeklinde listeler oluşturmaya izin veriyor.
- Yeni eklenenler, en çok satanlar şeklinde hazır bir liste önerisi sunuyor.
- Takas sonrası kullanıcılar birbirlerine puan veriyorlar.
- Açık kütüphanelerin veri tabanı sisteme yüklenerek kullanıcıların alması emanet sağlanıyor.

Olumlu Yorumlara Sahip Özelliklerin Tespit Edilmesi

Kullanıcıların memnuniyet durumlarını belirten en sık tekrarlanan yorumlar aşağıda belirtilmiştir:

- Elimde takas yapabileceğim kitaplar boş duracağına yerine başka kitap alabilmek güzel.
- Kargoyla elime ulaşması büyük kolaylık. Kargo takibi de çok kolay.
- Harita üzerinden kitabı görmek teslimat açısından çok iyi olmuş.
- Takas yapacağım kullanıcının takas puanına bakarak karar veriyorum.

Olumsuz Yorumların Değerlendirilmesi

Kullanıcıların memnun olmadıkları durumları belirten en sık tekrarlanan yorumlar aşağıda belirtilmiştir:

- IOS için yapılmış keşke ANDROID sürümü de olsaydı.
- Listeden kitap seçme özelliği güzel fakat takas etmek istediğim kitap listede yok. Ekleyemiyorum.
- Kitabın birden fazla fotoğrafını koyabilsem fiziksel durumu hakkında daha açıklayıcı olurdu.
- Mesajlaşma özelliğinde bazı kullanıcılar rahatsız ediyor. Engelleyebilme yok mu.
- Takas yapacağım kitaplar arasında fiyat farkı çok fazla. Karşılığında ücret talep etmek istiyorum. Ya da ucuz kitapla takas etmesinler diye önlem alınsın.
- Kitabın lokasyonu olarak kitabı eklediğin yer çıkmış. Kitap orada değil. Konumu değiştiremiyorum.

3.4 Kullanıcıların Temel İhtiyaçlarının Belirlenmesi

Temel ihtiyaçlar şunlar olarak belirlenmiştir.

- Kitabın paylaşılması

— Kitap Paylaşmak isteyen kullanıcı en fazla 3 adımda kitabı paylaşabilmeli.
— Paylaşılması istenen kitabın adı birkaç kelime ile yazıldıktan sonra kullanıcıya öneri olarak kitabın tam adı önerilmeli.

- Kitabın fotoğrafı ve içeriği hakkında kısa bilgi otomatik olarak görüntülenmeli.
- Kullanıcının eklemek istediği açıklama varsa yazabilmeli.
- Kullanıcı otomatik olarak önüne gelen kendi lokasyon bilgisini onaylamalı veya değiştirebilmeli.
- Paylaş butonu ile sistem kayıt yapılması gerekmektedir.

- Paylaşılan kitaplardan talep etme

- Paylaşım talep edecek kullanıcı öncelikle paylaşımdaki kitapların listesini görebilmeli.
- Kategoriye göre filtreleme yapılabilmeli.
- Kitap ismine göre ya da lokasyona göre sıralama yapılabilmeli.
- Kitabın adına veya içeriğindeki açıklamalara göre arama yapabilmeli.
- Kullanıcı istediği kitabı bulduktan sonra en fazla 2 adımda kitap talebi oluşturabilmelidir.
- Kitap talebi oluşturan kullanıcı paylaşan kişi ile irtibata geçmesi için özel bir mesajlaşma penceresinde haberleşmesi sağlanmalıdır.
- Bu aşamada anlaşan kişiler kitabı elden ya da kargo yoluyla aralarında belirledikleri yöntemle paylaşabilirler.

Asıl Amaca Uygun İhtiyaçlar

- Paylaşım yapmak ve elde etmek isteyenler arasında mesajlaşma imkânı sağlanması
- Harita üzerinde kitabın konumunun görünmesi
- Kargo ile paylaşımın gerçekleştirilebilmesi

Mobil Platforma Uygunluğu

Çok fazla veri girişi yapılmaması için kitabın özelliklerinin ve resminin otomatik olarak getirilmesi sağlanmalıdır.

Kararsız Kalınan Özellikler

Kitabı paylaşan kişi ücret talep etme ya da takas isteği asıl amaca uygun olup olmadığı tam belirlenememiştir. Bu nedenle şu aşamada bu özellik temel ihtiyaçlara eklenmemiştir.

3.5 Fark Yaratacak İhtiyaçlarının Belirlenmesi

Kullanıcının lokasyonuna göre en hızlı biçimde elde edebileceği kitapların önerilmesi sağlanmalıdır. Aynı zamanda kullanıcının paylaştığı kitaplara göre kendisine paylaşımda olan başka kitaplar da önerilmelidir. Bu özelliği gerçekleştirebilmek için bir öneri sistemi eklenmesine karar verilmiştir.

3.6 Gereksiz İhtiyaçlarının Belirlenmesi

Kitap yerine benzer özelliklere sahip olduğu düşünülen film dvd paylaşma özelliği eklemesi isteği bu uygulama için gereksiz görüldüğünden ihtiyaçlardan çıkartılmıştır.

3.7 Nötr İhtiyaçlarının Belirlenmesi

Paylaşılan kitaplardan değil de yeni bir kitabı yayınevinden satın alma ihtiyacı gereksiz olmamakla birlikte temel ihtiyaç olmaktan da uzaktır. Bu nedenle nötr ihtiyaç olarak belirlenmiştir. Böylece projenin kaynağı yettiği durumda değerlendirmeye alınmak üzere dokümanlara eklenmiştir.

3.8 Prototip Tasarımları Yardımıyla Kullanılabilirlik Testleri Yapılması

Bu aşamada belirlenen ihtiyaçları içeren bir prototip Adobe XD programı yardımıyla oluşturulmuştur. Bu oluşturulan prototip üzerinde paydaşların fikirleri doğrultusunda yukarıdaki adımlar tekrar değerlendirilmiştir.

4 Sonuçlar

Kullanıcılar istedikleri yerde ve anda erişebiliyor olmalarından dolayı mobil uygulamalara yönelmekte, gün geçtikçe daha fazla tercih etmektedirler. Mobil uygulamalarda gereksinimlerin net bir şekilde belirlenmesi projeleri başarıya götüren en önemli adım olarak karşımıza çıkmaktadır.

Bu çalışmada Kitap paylaşımı uygulaması örneği üzerinden önerilen gereksinim belirleme yöntemi uygulanmıştır. Prototip üzerinden uygulamayı kullanan kullanıcılara yapılan bir anket ile uygulamayı kullanıp kullanmayacakları sorulmuş ve %90 gibi bir sonuçla kullanabileceklerini belirtmişlerdir.

Aynı zamanda bu ankette kullanıcıların nerelerde zorlandıkları gözlemlenmiş ve temel ihtiyaçları karşılarken hiç zorlanmadıkları tespit edilmiştir. Bazı kullanıcılar bizim çıkartılmasını önerdiğimiz özelliğin eklenmesi gerektiğini ifade etmiştir. Ancak bu kullanıcıların sayısı %6'yı geçmemiştir. Bunun dışında da bu çalışmada hiçbir kategoride ele alınmayan yeni bazı özelliklerin de uygulamaya eklenmesi konusu da az sayıda kullanıcı tarafından bildirilmiştir. Bu özellikler ileride tekrar değerlendirmek üzere proje çalışma dokümana eklenmiştir.

Bu geri dönüşler ışığında bu çalışmada önerilen yöntemi kullanarak belirlenen gereksinimlerin başarılı olduğunu ifade edebiliriz.

Bundan sonraki çalışmalarda başka örnek mobil uygulamalarda bu yöntem kullanılarak başarısı ölçülebilir. Yöntem Mobil haricinde Web uygulamaları için geliştirilerek genelleştirilebilir.

Kaynakça

1. Ahimbisibwe, A., Cavana, R. Y., Daellenbach, U.: A contingency fit model of critical success factors for software development projects. Journal of Enterprise Information Management, vol. 28, pp. 7-33 (2015).
2. Dinçer, K., Garousi, V.: Yazılım Projelerinde Başarısızlık: Kritik Başarı Faktörlerine Dayalı bir Vaka Çalışması. Proceedings of the 9th Turkish National Software Engineering Symposium (UYMS 2015), s. 59–71. İzmir (2015)
3. Askaroglu, E., Yilmaz, H.: Uluslararası Geliştirilen Bir Projenin İşleyişi, Gözlemlerimiz ve Edinilen Tecrübeler. Proceedings of the 9th Turkish National Software Engineering Symposium (UYMS 2015), s. 447–453. İzmir (2015)
4. Akdur, D., Garousi, V., Demirörs, O. Gömülü Sistem Mühendisliğinde Kullanılan Yazılım Modellemesi ve Model Güdümlü Teknikler Anketi Türkiye Sonuçları. Proceedings of the 9th Turkish National Software Engineering Symposium (UYMS 2015) s.822-835 İzmir (2015)
5. Horouz IOS Uygulaması, https://apps.apple.com/us/app/horouz-kitap-takas-uygulamas%C4%B1/id1206012467, Erişim Tarihi: 01.06.2019
6. Kipu Mobil Uygulaması, https://play.google.com/store/apps/details?id=com.idea4mobile.projects.kipu, Erişim Tarihi: 01.06.2019
7. Kitapdostum Mobil Uygulaması, https://play.google.com/store/apps/details?id=com.kitapdostum&hl=tr, Erişim Tarihi: 01.06.2019
8. Kitapodam Web Uygulaması, http://www.kitapodam.com/, Erişim Tarihi: 01.06.2019
9. Albayrak, Ö.: Yazılım Mühendisligi Egitimi: Gereksinim Belirleme ve Analiz Asamasına Yönelik Deneyim ve Uygulamalar. Proceedings of III. Ulusal Yazılım Mühendisligi Sempozyumu ve Sergisi, s. 15-18, Ankara (2007)
10. Neczan, Ö., Özmen, T., Özer, P. S., Eriş, E. D.: Seçme Yazılarla Örgütler ve Yenilik. Ankara: Detay Yayıncılık. (2010)
11. Köklü, K.: İş Analizi, İş Analistliği ve İş Zekası. Lectio Socialis, Volume 2, Issue 2, s. 121-142, Temmuz (2018)
12. Giray G.: Bilgi Sistemlerinin Toplam Sahip Olma Maliyetini Hesaplamak İçin Bir Çerçeve. Bilişim Teknolojileri Dergisi, Cilt: 11, Sayi: 4, Ekim 2018
13. Kamu Kurumlarında Mobil Platformların Kullanımı, http://afyonluoglu.org/PublicWebFiles/Reports-TR/genel/201804-Kamu%20Kurumlar%C4%B1nda%20Mobil%20Platformlar%C4%B1n%20Kullan%C4%B1m%20Rehberi-TASLAK.pdf, Erişim Tarihi: 01.06.2019

Yazılım Etmenleri Kullanarak Çok Sayıda Zindan Bölümünün Yöntemsel Olarak Üretilmesi

Kaya Oğuz[0000−0002−1860−9127] ve Jane Lameski

İzmir Ekonomi Üniversitesi
Bilgisayar Mühendisliği Bölümü
Sakarya Cad. No:156, 35330, Balçova, İzmir
kaya.oguz@ieu.edu.tr, jane.lameski96@gmail.com

Özet. Yöntemsel içerik üretimi, oyun içeriğinin algoritmik olarak yaratılmasıdır. Zindanlar, rol yapma oyunlarında oyuncunun, oyuncu-olmayan karakterlerin ve yaratıkların çevreleriyle ve birbirleriyle etkileşime girebileceği kapalı ve sınırlı bir alan olarak sıklıkla kullanılırlar. Yapısal olarak zindanlar çeşitli boyutta odalardan ve bu odaları birbirine bağlayarak keşif ve gezinme sağlayan koridorlardan oluşurlar. Odaların temel seviye pozisyonları aynıdır ve odaların birbirleriyle olan bağlantıları zindan bölümü içindeki odalarla sınırlıdır. Bu zindan bölümleri çeşitli yöntemsel içerik üretimi algoritmaları ile üretilebilirler. Fakat, yazılım etmenleri bu konuda çok nadir kullanılmaktadır. Etmenlerin kullanımı, zindan üretimi sırasındaki işlerin tanımlanmasına, soyutlanmasına ve ayrık nesneler içinde kapsanmasına yardımcı olur. Ayrıca üretim süreci üzerinde daha fazla kontrol imkanı sunarlar. Bu çalışmada iki ana katkımız bulunmaktadır; odalar arasındaki bağlantıları oluştururken sadece odanın bulunduğu bölümdeki değil, zindandaki bütün odaları dikkate alıyoruz; zindanların ayrık bölümlerden değil, tüm bölümlerin bağlantılı olmasına dayanan ve yazılım etmenleri kullananan bir yöntem öneriyoruz.

Anahtar Kelimeler: Yöntemsel içerik üretimi, Yazılım etmenleri, Ayrık kümeler

Procedural Generation of Multiple Dungeon Levels Using Software Agents

Kaya Oğuz[0000−0002−1860−9127] and Jane Lameski

Izmir University of Economics
Department of Computer Engineering
Sakarya Cad. No:156, 35330, Balçova, Izmir, Turkey
kaya.oguz@ieu.edu.tr, jane.lameski96@gmail.com

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Abstract. Procedural content generation (PCG) is the algorithmic creation of game content. Dungeons are commonly used in role playing games (RPG) as closed and confined areas in which the player, the non-player characters (NPCs), and monsters can interact with the environment and with each other. Structurally, dungeons are made up of rooms of various sizes, connected with corridors that provide a means of exploration and navigation. The rooms have the same base level position and the connectedness of the rooms is limited to the rooms within the level. These levels can be generated by several PCG algorithms. However, software agents are rarely used in this regard. Using software agents helps the identification and isolation of concerns while encapsulating these concerns in discrete entities. It also gives greater control on the generation process. We have two main contributions; we consider the rooms on all levels while creating a dungeon of connected rooms; and we propose a method for the generation of levels using software agents for not discrete but connected levels of dungeons.

Keywords: Procedural content generation, Software agents, Disjoint sets

1 Introduction

Procedural content generation (PCG) is the go-to solution for developers to improve the replayability of their game while reducing the time to create the content manually. PCG has become commonplace because of the ever growing and competitive nature of the game industry where several games are published for personal computers, game consoles and mobile devices.

In its essence, PCG is the algorithmic creation of content. One of its early implementations can be found in games such as Rogue [25], and its derivatives, Diablo [4], in which the algorithm creates a new set of levels, or the so-called dungeons in Rogue-likes, along with items, monsters, treasures, and anything else that is required. While PCG is mostly used to create indoor [13] and outdoor levels [6], it has also been used to create quests [9,8], buildings [15], virtual worlds [22], cities [28,24], and stories [23]. A survey of PCG on all fronts is available [11], so are a textbook [20] and a more focused survey on procedural dungeon generation [13].

Dungeons are commonly used in role playing games (RPG) as closed and confined areas in which the player, the non-player characters (NPCs), and monsters can interact with the environment and with each other. Structurally, dungeons are made up of rooms of various sizes, connected with corridors that provide a means of exploration and navigation. In most cases, the dungeons are made up of separate levels that are connected via a group of connecting tiles, such as stairs or a passage that loads another dungeon level. Because of this single point of connection, it is relatively trivial to connect two or more separately generated but adjacent dungeon levels.

The floor plan of a dungeon level is basically a two-dimensional representation while a three-dimensional representation can be used in the game. The rooms

have the same base level position and the connectedness of the rooms is limited to the rooms within the level. In this study, we have extended this approach to three-dimensions, while maintaining the trivial structural generation of the level.

There are several approaches that are commonly used in procedural dungeon generation, and they will be discussed in the following section. However, software agents, entities that can sense their environment and act accordingly [18], are rarely mentioned as a method. Using software agents in dungeon generation helps the identification and isolation of concerns while encapsulating these concerns in discrete entities. It also gives greater control on the generation process. Although the purpose of the PCG algorithm is to minimize human intervention in the generation of the content, a set of parameters to control the output is always welcome. Our study contributes to the existing literature by proposing a method that uses software agents that consider all rooms in all levels for connectivity during the generation of dungeon levels.

The remaining paper is structured as follows. The following section discusses the existing approaches to generate dungeons. Our approach which incorporates software agents to generate levels that are connected in three dimensions, without the limitations of a single level generation process is detailed in Section 3. The results are given in Section 4. The paper concludes with Section 5 which discusses the results and the future work.

2 Related Work

Dungeon or level generation is one of the most studied and commonly used branches of PCG algorithms. The purpose of this study is not to cover all of them; therefore the foremost methods will be mentioned. Interested readers are kindly directed to the relatively recent survey on procedural dungeon generation by Linden et al. [13] as a starting point.

A typical straightforward yet effective approach to generating dungeon levels is to place rooms randomly and connect them with corridors. Nystrom places randomly generated rooms in a confined space and then connects the rooms via a maze-like approach [16]. Another approach is to use space partitioning, such as binary space partitioning and store the connections in a space-partitioning tree so that the connectivity of the rooms can be arranged [19].

Although it does not have the structure of a typical dungeon, cellular automata have been used to generate levels with an organic and natural feel, rather than a man-made synthetic feel [12,1]. Cellular automata are the application of a set of rules to a grid of cells that have states that change according to these rules. Cellular automata is mostly associated with Conway's "Game of Life" [10], but they can also be used as a method that have very few parameters and a loose control on the output of an organic cave-like indoor levels.

While they are mostly used to create language and code parsers, generative grammars can be used generate game levels if the alphabet of the grammar refers to specific parts of the level, and the rules refer to the ways these parts can be

combined [9]. Dormans used generative grammars to combine the generation of missions with the generation of levels to create the two vital but highly coupled parts of the gameplay. Grammars have become a common approach to generate levels, such as the bachelor thesis of Adams [2] and the dungeon crawler by Linden et al.[14].

Evolutionary methods, such as genetic algorithms, are also popular in the procedural generation of dungeons. Genetic algorithms require the representation of the level as a chromosome, and a fitness function to evaluate the randomly generated chromosomes. Depending on the outcome of the fitness function, chromosomes can be used to produce new chromosomes, until a satisfactory fitness value is reached. Valtchanov and Brown use a tree structure for the representation of the level, and a fitness function that favors tightly packed clusters of rooms that have efficient connections [26]. Other studies are by Ashlock et al. which focuses on maze like levels [3], by Brown et al. in which they apply evolutionary algorithms to generate levels for the recently popular game Hotline Miami [5], and by Ruela and Delgado who also have worked on graph-based procedural content generation [17].

Software agents have been used to generate terrains [7] and 2D platformers [27], both of which have significantly different structures than dungeons. An application of software agents to procedural level generation is given in the third chapter of PCG book by Shaker et al. [19]. This approach uses an agent that digs tunnels and create rooms sequentially. Recently, Sheffield and Shah have improved this approach by incorporating reinforcement learning algorithm to the agent [21].

3 Proposed Method

There are several advantages for using software agents in the procedural generation of dungeons. The foremost advantage is the separation of concerns by encapsulating each to a single discrete entity. For each of these agents with different concerns, a different set of parameters could be given, improving the control of the developer over the creation process. It is also possible to switch agents with different approaches for the same problem.

With all these advantages in mind, we have come up with several agent types that operate on different parts of the dungeon with a loose hierarchical system. As shown in Figure 1, the dungeon agent is at the top, because it calls other agents sequentially to create the dungeon.

3.1 Dungeon Agent

The dungeon agent (DA) is responsible for the generation of dungeon levels, therefore it accepts parameters for the number of these levels, n, the width w and height h of each floor, space size limit s which depends on w and h, and probability value p for partitioning probability, b for blocking some of the regions for variety.

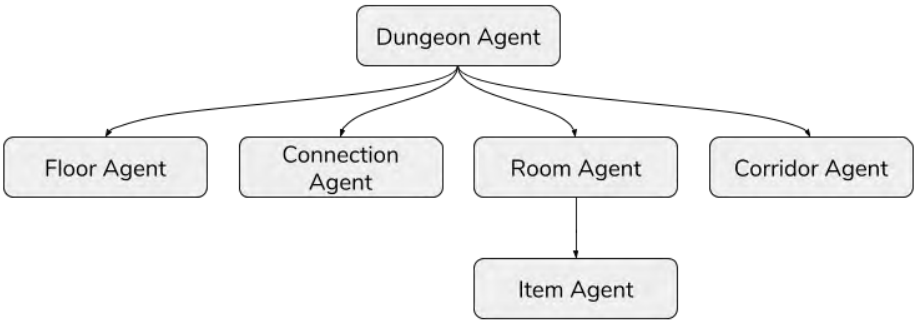


Fig. 1. The loose hierarchical view of the agents. The level agent calls for the main agents to create the structure of the dungeon, while the room agent calls item agent to create and position items in rooms.

The agent creates a space of size $w \times h$ and passes w, h, s, p and b to n number of spawned floor agents (FA). The partitioned spaces from all floors are retrieved and DA spawns a room agent (RA) for each of these spaces. DA then creates a connection agent (CA) and passes the spaces to retrieve the connections and spawns a corridor agent (CRA) and passes the rooms and connections to retrieve the corridors that connect these rooms. While this is the general flow of DA, each agent works on its own by considering the entities created by other agents.

3.2 Floor Agent

The floor agent is responsible for partitioning the space for each individual room. It uses a simplified binary space partition algorithm which randomly divides the defined area to two parts and continues to work on these parts until a threshold is reached. The spaces are not saved in a binary tree, because their connection will be handled by another agent. The flow of FA is detailed in Algorithm 1. The listing contains utility functions such as `Area` and `RandomValue` which would be available on any programming language library, or could be implemented trivially.

As the Algorithm 1 shows, FA defines a simplified binary space partition method and uses it recursively on generated spaces. If the area of current space is greater than the `SpaceLimit`, then the method finds the longer edge of the space. It randomly selects a point `T` on that edge to create two subspaces, `S1` and `S2`. FA rolls the dice for partitioning these spaces further by using the simplified binary space partitioning method recursively or stop and add the subspace to `L`. At this point the probability value `p` is reduced to `p x p`, making it more likely to partition the space earlier in the recursive calls. This is a preference that can be reversed, since we favored larger rooms to be available in the generated levels.

Algorithm 1: The Floor Agent uses a simplified binary space partitioning approach to generate spaces.

```
Input  :  w, h, s, b, p
Output :  L

Create space S at point (0, 0) with width w, height h
Create L, an empty list for generated spaces
SpaceLimit = Area(S) x s
Define and call SimplifiedBinarySpacePartition(S, p):
    If Area(S) > SpaceLimit:
        Find the longer edge of S, either its width or length
        Randomly select a point T on that edge
        Use T to create subspaces S1, S2
        If RandomValue() > p:
            SimplifiedBinarySpacePartition(S1, p x p)
        Else:
            Append S1 into list L
        If RandomValue() > p:
            SimplifiedBinarySpacePartition(S2, p x p)
        Else:
            Append S2 into list L
    Else:
        If RandomValue() > p:
            Append S into list L
L contains a list of spaces
```

If the area of the space `S` is not greater than the `SpaceLimit`, then the method checks the blocking probability and adds `S` to `L`.

3.3 Room and Item Agents

For each space generated by the floor agent, dungeon agent spawns a room agent which works on the space given to it by generating a room that can fit into it. It then spawns an item agent (IA) which randomly places items on the floors of these rooms. IA receives a parameter m which limits the number of items it can generate for a room.

The agents for room and items are implemented more straightforward than other agents, although they can include a wide range of options depending on the context of the game. In this case, the focus was on the demonstration of agents, therefore a rectangular area and random spots on the floors were all that was needed.

3.4 Connection and Corridor Agents

Connection agent is vital for the generation of connected rooms among all levels of the dungeon. CA is spawned when the FA are done. It receives the spaces on

Algorithm 2: The Connection Agent uses disjoint sets a bounding box collision detection to locate the adjacent spaces.

```
Input  :  List of spaces L
Output :  List of connected spaces C

For each space S in L:
    For each space T in L - {S}:
        If S and T are on the same level:
            If S and T share an edge:
                Set T as a neighbor of S
        Else:
            If T is on the level above:
                If Bounding Box Collision detects an overlap:
                    Add T as a neighbor of S
                    Add S as a neighbor of T

For each space S in L:
    If S has no neighbors:
        Remove S from L

Let C be the list of connections
Set all spaces to a set of their own
Until there is only one set left:
    Randomly pick S from L
    Randomly pick N from neighbors of S
    If S and N are NOT in the same set:
        Find the set of S, S1
        Find the set of N, S2
        Perform a union operation on S1 and S2
        Store the connection between S and N in C
```

all levels and finds the adjacent spaces on the same level, or the levels above or below. For the spaces on the same floor, it looks for the edges that are shared among these spaces to define them as neighbors. For the spaces on different floors, it uses a simple bounding box collision check to determine if two spaces overlap.

The flow of CA is given in Algorithm 2. As the algorithm shows, the connection agent first determines the neighbors of each space. This is done by going through the list of all spaces and comparing them to the list of remaining spaces. If two spaces, `S` and `T` are on the same level, then the agent checks if they share an edge. To do so, it runs a list of checks on the top left corner of the spaces, along with their width and height. If they share an edge, then `T` is set as a neighbor of `S`. The neighbors are stored as a list, so a space can have many neighbors. If these spaces are not on the same floor, then it is possible that they overlap.

Bounding Box Collision can be used to find overlapping spaces and set them as neighbors.

At this point, there might be some spaces that don't have any neighbors because of the blocked spaces. Connection agent finds and removes them from the list of spaces.

In a regular level, all rooms are expected to be connected, providing accessibility to each. In our study, the rooms on all floors are considered for connection. This creates rooms that are inaccessible from their own floor, but another path through other rooms on other floors is generated. To make sure that all rooms are connected, the final part of the CA in Algorithm 2 uses the disjoint set approach with the union and find methods. It initializes all spaces on a set of their own, and continues to apply the union function to the sets until there is only one set left. It does so by first randomly picking a space `S` from the list of spaces `L`, and randomly picking one of its neighbors `N`. If these spaces are not in the same set, then they are joined by the union operation. Also, the connection between them are stored in the list of connections `C`.

Once the CA is done, the disjoint set algorithm ensures the connectedness of all rooms. These connections, along with the list of rooms, are passed to the corridor agent. The corridor agent operates on the rooms in the connected spaces. For each room `R`, it finds the room `Q` in the connected space. If they are on the same floor, it finds their relative positions and draws a corridor from `R` to `Q`. If they are on separate floors, it assigns a space in both rooms as stairs that connects them together.

As with the room and item agents, the corridor agent is designed to serve its purpose for the demonstration of agents. Several different approaches can be used to extend its capabilities; such as using path finding algorithms to build the corridors.

4 Results and Discussion

The time complexity of the algorithm is $O(n^2)$ where n is the number of spaces generated in all levels. The algorithm compares each space with the other ones in order to find the neighbors. Since the number of spaces are moderately low, the agents complete their work within seconds. On the other hand, the generation is usually done offline, so the current time complexity of the algorithm is well within accepted values.

The agents are implemented using Python and the resulting maps are generated as image files. A sample output of the three levels of a dungeon with ten levels is given in Figure 2. The parameters were; 1024×1024 is given as the size of each level, room size limit is set as the 0.05 percent of the total area, space partitioning probability is set to 0.3 and the room block probability is set to 0.05. To keep the maps simpler, the items are not printed on the images. Since the outputs are in two-dimensions, the stairs are shown as gray squares inside the rooms. As the maps show, there are some rooms that are not accessible from the rooms within the level itself, however the connections to the other rooms

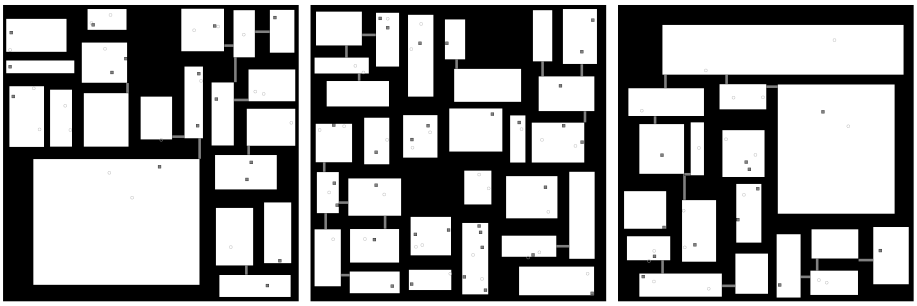


Fig. 2. Three floors of ten dungeon levels generated by the agents.

above or below the level provide access to them. The effect of space partitioning can be seen in the level on the left and right, while the room blocking probability can be observed in the level on the left and the level in the middle.

The levels generated by this study look similar to the ones in the existing literature. However, we introduce the multi-level connectivity of the rooms and the realization of several software agents to generate the level. While we put forward and implement these contributions, we would also like to discuss the apparent improvements and existing weaknesses in this section.

Since the agents are kept relatively simple and straightforward in this preliminary study, they were able to work in harmony without any apparent conflicting issues in their linear fashion. The agents could be improved by adding variety to their behaviours. A typical example would be incorporating random turns in long corridors for the corridor agent, and using geometric shapes other than rectangles for the room agents. More agent types could also be added; the room agent could make use of more agents in their design, such as the inclusion of an agent that can handle the lighting of the rooms, another one to position objects.

The agents could be redesigned so that they could be started simultaneously. However currently this would not produce good results because in their current form because the connection agent would require the generation of all rooms to create paths that make use of rooms on other levels.

5 Conclusion and Future Work

We have defined a set of agents to handle the procedural generation of a dungeon. Most dungeons provide several dungeon levels, each of which are isolated from the remaining levels except their single point connections to the levels just above or below them. The rooms on each level are connected and accessible within the level. Besides using agents, our solution considers the rooms in all levels at the same time and all of them for creating a connected set of rooms, on multiple floors.

The lack of agents on the procedural content generation front is surprising, because they provide several advantages; such as the separation of concerns and the ability to replace agents with ones that have other purposes if need be.

The agents appear to generate the levels rather quickly even on an interpreter such as Python. A test run which included 100 levels, with room size limit and room block probability both set to 0.05, is completed within seconds. Compared to other agent based solutions our method provides more control on the generation process because of using agents for each concern.

This preliminary work is planned to be ported to a game engine, such as Unity, so that the dungeons can be immediately viewed in three dimensions, along with their textures and objects. This will also allow us to create more complicated agents that could consider more cosmetic challenges, such as the placement of lights and objects in the rooms. From an object oriented point of view, agents are also planned to define specific interfaces so that they could easily be extended.

References

1. Adams, C., Parekh, H., Louis, S.J.: Procedural level design using an interactive cellular automata genetic algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. pp. 85–86. GECCO '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3067695.3075614
2. Adams, D.: Automatic generation of dungeons for computer games. Bachelor thesis, University of Sheffield (may 2002)
3. Ashlock, D., Lee, C., McGuinness, C.: Simultaneous dual level creation for games. IEEE Computational Intelligence Magazine **6**(2), 26–37 (May 2011). https://doi.org/10.1109/MCI.2011.940622
4. Blizzard North: Diablo. Digital Game (December 1996)
5. Brown, J.A., Lutfullin, B., Oreshin, P.: Procedural content generation of level layouts for hotline miami. In: 2017 9th Computer Science and Electronic Engineering (CEEC). pp. 106–111 (Sep 2017). https://doi.org/10.1109/CEEC.2017.8101608
6. Doran, J., Parberry, I.: Controlled procedural terrain generation using software agents. IEEE Transactions on Computational Intelligence and AI in Games **2**(2), 111–119 (June 2010). https://doi.org/10.1109/TCIAIG.2010.2049020
7. Doran, J., Parberry, I.: Controlled procedural terrain generation using software agents. IEEE Transactions on Computational Intelligence and AI in Games **2**(2), 111–119 (June 2010). https://doi.org/10.1109/TCIAIG.2010.2049020
8. Doran, J., Parberry, I.: A prototype quest generator based on a structural analysis of quests from four mmorpgs. In: Proceedings of the 2Nd International Workshop on Procedural Content Generation in Games. pp. 1:1–1:8. PCGames '11, ACM, New York, NY, USA (2011). https://doi.org/10.1145/2000919.2000920
9. Dormans, J.: Adventures in level design: Generating missions and spaces for action adventure games. In: Proceedings of the 2010 Workshop on Procedural Content Generation in Games. pp. 1:1–1:8. PCGames '10, ACM, New York, NY, USA (2010). https://doi.org/10.1145/1814256.1814257
10. Gardner, M.: Mathematical games: The fantastic combinations of John Conway's new solitaire game 'life'. Scientific American **223**, 120–123 (oct 1970)

11. Hendrikx, M., Meijer, S., Van Der Velden, J., Iosup, A.: Procedural content generation for games: A survey. ACM Trans. Multimedia Comput. Commun. Appl. **9**(1), 1:1–1:22 (Feb 2013). https://doi.org/10.1145/2422956.2422957
12. Johnson, L., Yannakakis, G.N., Togelius, J.: Cellular automata for real-time generation of infinite cave levels. In: Proceedings of the 2010 Workshop on Procedural Content Generation in Games. pp. 10:1–10:4. PCGames '10, ACM, New York, NY, USA (2010). https://doi.org/10.1145/1814256.1814266
13. Van der Linden, R., Lopes, R., Bidarra, R.: Procedural generation of dungeons. IEEE Transactions on Computational Intelligence and AI in Games **6**(1), 78–89 (March 2014). https://doi.org/10.1109/TCIAIG.2013.2290371
14. Van der Linden, R., Lopes, R., Bidarra, R.: Designing procedurally generated levels. In: Ninth Artificial Intelligence and Interactive Digital Entertainment Conference (2013)
15. Müller, P., Wonka, P., Haegler, S., Ulmer, A., Van Gool, L.: Procedural modeling of buildings. ACM Trans. Graph. **25**(3), 614–623 (Jul 2006). https://doi.org/10.1145/1141911.1141931
16. Nystrom, B.: Rooms and mazes: A procedural dungeon generator. Online (dec 2014), https://journal.stuffwithstuff.com/2014/12/21/rooms-and-mazes/
17. Ruela, A.S., Valdivia Delgado, K.: Scale-free evolutionary level generation. In: 2018 IEEE Conference on Computational Intelligence and Games (CIG). pp. 1–8 (Aug 2018). https://doi.org/10.1109/CIG.2018.8490366
18. Russell, S., Norvig, P.: Artificial Intelligence, A Modern Approach. Prentice Hall, third edn. (2010)
19. Shaker, N., Liapis, A., Togelius, J., Lopes, R., Bidarra, R.: Constructive generation methods for dungeons and levels. In: Shaker, N., Togelius, J., Nelson, M.J. (eds.) Procedural Content Generation in Games: A Textbook and an Overview of Current Research, pp. 31–55. Springer (2016)
20. Shaker, N., Togelius, J., Nelson, M.J.: Procedural Content Generation in Games: A Textbook and an Overview of Current Research. Springer (2016)
21. Sheffield, E.C., Shah, M.D.: Dungeon digger: Apprenticeship learning for procedural dungeon building agents. In: Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts. pp. 603–610. CHI PLAY '18 Extended Abstracts, ACM, New York, NY, USA (2018). https://doi.org/10.1145/3270316.3271539
22. Smelik, R., Tutenel, T., de Kraker, K., Bidarra, R.: A declarative approach to procedural modeling of virtual worlds. Computers & Graphics **35**(2), 352 – 363 (2011). https://doi.org/http://dx.doi.org/10.1016/j.cag.2010.11.011, virtual Reality in Brazil Visual Computing in Biology and Medicine Semantic 3D media and content Cultural Heritage
23. Tearse, B., Mawhorter, P., Mateas, M., Wardrip-Fruin, N.: Skald: Minstrel reconstructed. IEEE Transactions on Computational Intelligence and AI in Games **6**(2), 156–165 (June 2014). https://doi.org/10.1109/TCIAIG.2013.2292313
24. Temucin, M.B., Oguz, K.: Procedural city generation using cellular automata. In: Eurasiagraphics 2017 Conference Proceedings. pp. 27–41 (2018)
25. Toy, M., Wichman, G., Arnold, K., Lane, J.: Rogue. Digital Game (1980)
26. Valtchanov, V., Brown, J.A.: Evolving dungeon crawler levels with relative placement. In: Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering. pp. 27–35. C3S2E '12, ACM, New York, NY, USA (2012). https://doi.org/10.1145/2347583.2347587

27. Wheat, D., Masek, M., Lam, C.P., Hingston, P.: Dynamic difficulty adjustment in 2d platformers through agent-based procedural level generation. In: 2015 IEEE International Conference on Systems, Man, and Cybernetics. pp. 2778–2785 (Oct 2015). https://doi.org/10.1109/SMC.2015.485
28. Whelan, G., Kelly, G., McCabe, H.: Roll your own city. In: Proceedings of the 3rd International Conference on Digital Interactive Media in Entertainment and Arts. pp. 534–535. DIMEA '08, ACM, New York, NY, USA (2008). https://doi.org/10.1145/1413634.1413742

Blok Zinciri Kullanarak Tedarik Zinciri Yönetimi

Hazal Onay, İlayda Cansın Koç, Kuthan Temimhan, Hakan Yazıcı[1] and Murat Erten[2]

[1] İzmir Yüksek Teknoloji Enstitüsü
[2] Bakırçay Universitesi
{hazalonay,ilaydakoc}@std.iyte.edu.tr;{murat.erten}@bakircay.edu.tr

Özet. Bu çalışmada blok zinciri tabanlı bir tedarik zinciri yönetim sistemi geliştirilmiştir. Sistem özellikle çevre duyarlı ürünler için uygun olup, bir alışveriş sitesi içermektedir. Buraya verilen siparişler ile ilgili olarak satıcı ve alıcı arasında ethereum tabanlı bir akıllı kontrat oluşturulmaktadır. Bu kontratın ardından satıcı herhangi bir kargo şirketiyle ayrı bir akıllı kontrat oluşturmaktadır. Bu ikinci kontrat sıcaklık, nem gibi taşıma koşullarını da içermektedir. Taşıma işlemi boyunca bu değerler sensörler aracılığıyla okunarak kaydedilmekte ve özel bir blok zincirine kaydedilerek tarafların erişimine sunulmaktadr. Ürünlerin teslim edilmesi halinde alıcı bu değerleri kontrol etmekte, başta belirlenen şartlar sağlanmışsa ödemeyi yapmaktadır. Aksi halde alıcının kontratı iptal etme ya da bir indirim alma hakkı doğmaktadr. Alıcının teslimatı onaylamasının ardından satıcı nakliye şirketine ödemeyi yapabilmektedir.

Anahtar Kelimeler: Blok zinciri, Ethereum, Tedarik zinciri yönetimi.

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Supply Chain Management Using Blockchain

Hazal Onay, İlayda Cansın Koç, Kuthan Temimhan, Hakan Yazıcı[1] and Murat Erten[2]

[1] Izmir Institute of Technology
[2] Bakırçay University
{hazalonay,ilaydakoc}@std.iyte.edu.tr;{murat.erten}@bakircay.edu.tr

Abstract. In this study a supply chain management system was developed using blockchain technology. The system, which may be suitable to environment sensitive products involves a simple market place, where a smart contract is introduced between the seller and the buyer using ethereum when an order is placed. Upon completion of this contract a different smart contract is established between the seller and the shipping company. This latter contract involves the shipping conditions such as temperature and humidity of the transport medium. These data are recorded using a sensor system in the container and kept in a private blockchain but are available to all the parties involved. Upon delivery of the goods, the buyers approves payment to the supplier if the conditions set for delivery were complied with. Then the seller approves the payment to the shipper and the file is closed. All the transactions are performed using Ether and the private blockchain assures that the shipper has honoured the conditions set forth in the contract.

Keywords: Blockchain, Ethereum, Supply cahin management.

1 Introduction

In supply chain management, how products, especially agricultural products, are transported has great importance. The buyer wants to know under what circumstances the products he/she buys are transported and which stages they have gone through. The conditions under which the products are transported such as temperature or humidity as well as the location could be recorded periodically to assure the buyer that he will receive the goods as expected. If there is something wrong in the data like exceeded temperature value, this data should be available to third parties. In this respect, block chain may play a big role to control the circumstances.

Although blockchain technology was developed for Bitcoin, a crypto currency, it has become one of the major future technologies that would reshape many areas and industries. Reliable collection and protection of the data is very important for many industries. Altering/changing data (deleting security camera records, modifying important data in companys favor) is one of today's main

problems. In such cases, it is safer to keep records in the blockchain, instead of keeping the records in a single database, i.e. on hard discs or cloud systems. Through the transparency provided by the blockchain, fraudulent actions may be reduced and the trust between parties will be increased. Also, if we think of a supply chain scenario which involves the transportation of fresh vegetables, fruits or dairy products, it enhances the food safety. There is also less food wastage because every single piece of food is accounted for and it also provides the customers trustful information about their food and empowers them to make better buying choices.

Blockchain can also be used to introduce smart contracts for trade operations between buyers and sellers. An order where a smart contract sets the conditions introduces more reliability into the process. When an order is placed, the seller needs to be sure that he/she will receive the payment, and the buyers would like to be sure that they will receive what they have asked for. There is also a shipping company involved in such transactions and their payments could also be governed by smart contracts. This approach also allows one to make the transfer of payments faster. It will increase the processing speed, and in most cases, eliminate the involvement of third parties such as banks, thereby reducing costs. The main contribution of blockchain to data integrity is that generated transaction can only be read and only new transactions can be added to the block chain. Altering them would be extremely difficult.

In this study, a system which offers a block chain solution to supply chain management, which also answers the above requirements is proposed. On the one hand, a smart contract is established between the seller and the buyer using ethereum block chain network, and all the financial transactions are carried out using this system. On the other hand the shipment conditions are monitored continuously and recorded to a private block chain to make sure that all the parties involved in the sales process, that is the seller and the shipper, comply with the requirements stated in the contract.

In the proceeding section similar studies and commercial applications in this area will be introduced. In section 3 the proposed system will be described. Discussions are included in section 4.

2 Related Work

Use of blockchain in food logistics is an important issue studied by many academic and commercial gorups. In [1] the author proposed an RFID based system where blockchain technology is utilized. In [2], the authors reviewed the current status of blockchain technology. They also investigated potential use of this technology in manufacturing and they proposed a future blockchain ready manufacturing supply chain. Article referenced in [3] studied both the transaction side and the logistic side, similar to our study, of adoption of blockchain in logistics industry. Another study on this subject is “Smart Contracts and their Application in Supply Chain Management“ which is a thesis carried out at MIT where supply chain is attempted to be improved using smart contract [4].

Supply chain management using block chain has also found many commercial applications which involve Internet of Things (IoT). IOTA is the the first open-source distributed ledger that is being built to power the future of the Internet of Things with feeless micro transactions and data integrity for machines [5]. IoT Chain(ITC) is developed as a lite operating system using the block chain concept allowing data to be layered and stored in a decentralized manner and providing protection with the combined strength of the millions of IoT nodes within the network[6]. Atonomi provides IoT developers and manufacturers with an embedded solution to secure devices with blockchain-based immutable identity and reputation tracking [7].

In supply chain management side a similar work is the implementation of Coffee Supply Chain using Smart Contract developed by ImperialSoftech [8]. Walmart has chosen IBM Food Trust, which has been developed for consumer companies, like Dole, Wegmans and Unilever, to track products moving through the supply chain using block chain technology. During the transportation process, manual entry is made on the blockchain for receiveing and handing over the goods. Maersk Line, which is a maritime transport company, has been using blockchain technology since 2017 for tracking and managing its cargo. In this system, the company, Dutch customs and US Homeland Security could access data on the cargo. The shipping company UPS(United Parcel Service), together with the Blockchain in Trucking Alliance (BiTA) is working to develop blockchain standards for the freight industry [9].

The main difference between the work described above and our proposed system is that our project operates as an e-commerce site. The other thing is, ether cryptocurrency is used for transactions.

3 The proposed system

In our proposed system the smart contract involves the buyer, the seller and the shipper. Upon receiving an order, the seller creates a smart contract, and this transaction is transmitted in the ethereum network and added to a block. The seller then generates another transaction with the shipper, indication the shipping conditions, such as the container temperature and humidity. This transaction is also recorded into a block, again in the ethereum network.

During the transportation process the conditions in the container are recorded by the sensors periodically and these values are kept in a private block chain. Once the buyer receives the goods, he/she can check the private block chain through the user interface provided and if convinced that the goods have been transported as promised, approves the transaction to the seller. The seller, having received the payment as Ethers, approves the payment to the shipper, again as Ethers. If, however, the temperature or humidity exceeded the required values during transportation, then the buyer may cancel the order or he/she may be offered a discount on the contract.

The proposed system has two main components: the smart contracts implemented on ethereum, and the container monitoring system which has been

implemented using a private block chain. The block diagrams of both systems are shown in Fig. 1, Fig. 2 and Fig. 3 .

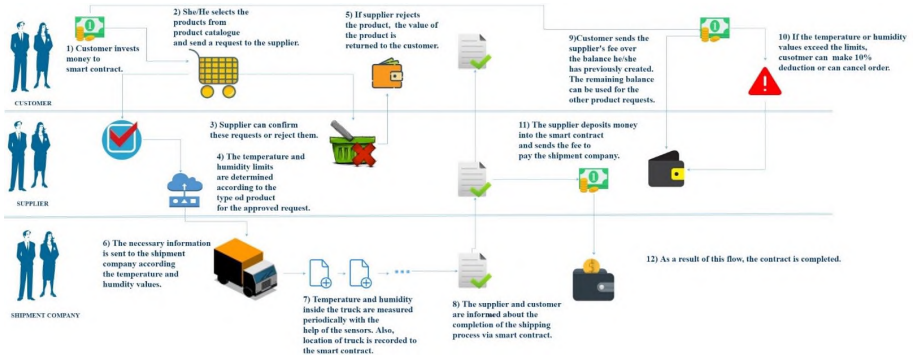


Fig. 1. The smart contract

Ethereum block chain which is used to keep records and the smart contract is written using *solidity* language and implemented on website. In this project, *Metamask* is used for deploying smart contract, exchanging payment between supplier (seller and supplier have been used interchangeably, as well as the customer and buyer throughout the text) and customer; and, supplier can also send shipper fee with this tool. In other words, it acts as a bank account. Also, Ether crypto currency is used to exchange funds.

The structure of supply chain smart contract in our project is as follows:

- The customer sends the request to the seller to buy the product(s) and starts the smart contract.
- Customer should also contribute to the contract to prove that she/he has funds for payment to the supplier.
- Supplier can see this request. If supplier approves the requests, he/she creates another smart contract with shipper to initiate shipment.
- If supplier denies the requests, customer is informed immediately, and product fee is refunded to the customer via Metamask.
- With shipment contract, supplier should define lower and upper limits of temperature and humidity
- These values are read from the sensors in the container periodically and automatically saved to the private block chain.
- When everything is completed, customer finalizes the smart contract and transfers the money to supplier. This transfer is provided by Metamask using ether virtual money.
- After customer makes the payment, supplier should send the fee to shipper.
- If the sensor limits are exceeded during the transportation process, customer can deduct a pre-set amount from supplier or cancel the order. Supplier my then apply the same process to the shipment company.

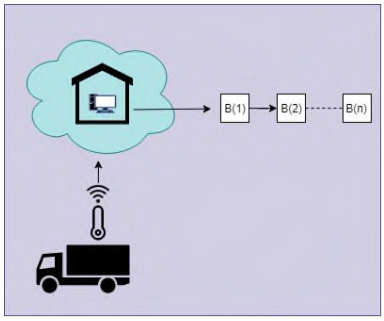


Fig. 2. The private block chain located at the container

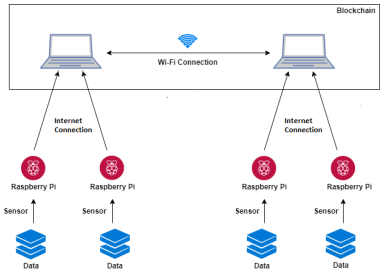


Fig. 3. The sensors and the private blockchain

In our proposed solution we employed data collector sensors and a controller (Raspberry Pi) at the cargo section of the truck in order to collect temperature and humidity data. This data is then recorded in a blockchain. The sensor board DHT11 is used for measuring the temperature and humidity. The user interface designed allows the supplier to keep track of the orders from different customers, and the customers are also provided with a user interface where they can access the instantaneous values read from the transportation medium. This way, all of the data will be transparently available to anyone. So, any party involved in this transportation process will take advantage of using block chain as opposed to using traditional database systems.

Each transaction is mined in nearly 0.25 seconds in the prototype. Also, the amount of gas used for each transaction is almost 0.000021 Ether. The performance of the prototype is limited by the Ethereum network capacity. If the number of Ethereum blocks increase, the mining time is also increased.

4 Discussions and Conclusions

In this study, an Ethereum based smart contract system and a private block chain based IoT solution is described. The smart contract protects the payments to be received and paid by the seller and buyer respectively. The IoT system makes sure that the goods are transported as agreed. The smart contracts is integrated with the user interface and presented to the user as e-commerce site. The transfer step of the product from the vendor to the buyer is recorded in a private block chain. The changes that the products have undergone during the transfer phase have been presented to all parties simultaneously. In the payments, banks were removed and shopping with crypto currency was provided.

Through using block chain:

– The need for a central server is eliminated: the data stored in the blockchain is shared by everyone who is part of blockchain network.

– The data stored in blockchain is immutable since it is protected by cryptographic hash functions. Merkle Tree implementation is how we check the integrity of the data.
– Every single operation (in our case, the temperature and humidity data collection) in the supply chain will be recorded in the blockchain transparently. So anyone who is part of the blockchain network will be able to see these data. Everyone will, therefore, be forced to be accountable for their actions.
– If there is significant change in the temperature and/or humidity data, the manager of operations will be able to see the change through the web immediately and take an action accordingly to avoid any problem before it grows bigger.

It is generally desired to have low latency in the IoT devices, however, block mining is time consuming operation that is caused by the encryption algorithms that run to keep the data safe. This is one of the issues related to our proposed system which needs further study to be solved. In the future, instead of ether cryptocurrency, a new cryptocurrency can be devised. The speed of mining is expected to decrease as the number of blocks is reduced.

There is a risk of loosing WiFi connection which can be an important problem. In this case, local storage devices must be used in each container in order to store the generated data to prevent loss.

There is also the possibility of the devices in the transportation vehicle being hacked hence sending faulty, fraudulent data. Possible solutions to this problem could be only making necessary ports available and not allowing the services to accept transactions coming form other ports. Also, there are still some open questions about how we can establish distributed trust among parties in the presence of vulnerabilities to Denial of Service(DoS) attacks, 51

In conclusion, in our proposed solution, we tried to introduce a supply chain management system using blockchain technology with its basic features. With certain changes and modifications, our proposed solution can accomodate to new technologies such as artificial intelligence and machine learning.

As future work, solutions to the current shortcomings will be searched and a mobile application for accessing the data and performing similar operations like the web page will be developed.

References

1. Tian, F., "An agri-food supply chain traceability system for China based on RFID and blockchain technology," 2016 13th International Conference on Service Systems and Service Management (ICSSSM),pp. 1-6, Kunming, 2016.
2. Abeyratne, S.,Monfared,R. P., "A Blockchain ready manufacturing supply chain using distributed ledger", https://dspace.lboro.ac.uk/2134/22625, (accessed June 4, 2019)
3. Francisco, K., Swanson,D., "The Supply Chain Has No Clothes: Technology Adoption of Blockchain for Supply Chain Transparency", Logistics, vol. 2, no:1, (2018).
4. MIT, "Smart Contracts and their Application in Supply Chain Management", https://dspace.mit.edu/handle/1721.1/114082 (Accessed: June 19, 2019).

5. IOTA Homepage, “The Next Generation of Distributed Ledger“, Available at: https://www.iota.org/ (Accessed: May 28, 2019).
6. IoT Chain Homepage, “IoT Chain - A High-Security Lite IoT OS - ITC“, Available at: https://iotchain.io/ (Accessed: May 28, 2019).
7. Atonomi Homepage, “Bringing Trust and Security To Iot“, Available at: https://atonomi.io/ (Accessed: May 28, 2019).
8. Gthub “Implementation of Coffee Supply chain using Ethereum smart contract“ https://github.com/imperialsoftech/coffee-supplychain-ethereum.
9. Techrepublic “5 companies using blockchain to drive their supply chain“, https://www.techrepublic.com/article/5-companies-using-blockchain-to-drive-their-supply-chain/.

Yazılım Proje Yaşam Döngüsü Süreç Metrikleri ile Proje Performanslarının Ölçülmesi ve Proje Başarı Puanının Hesaplanmasına Yönelik Bir Model Geliştirilmesi

Özgür Gün[1], Pınar Yıldız Kumru[2]

[1,2] Endüstri Mühendisliği Bölümü, Kocaeli Üniversitesi, Kocaeli, Türkiye
ozgur.gun@tubitak.gov.tr, pinarki@kocaeli.edu.tr

Özet. Bu tez çalışmasında sistem ve yazılım projelerinin performanslarını proje yaşam döngüsü süreç metriklerini kullanarak değerlendiren ve bu değerlendirme sonucu bir proje performans başarı puanı hesaplayan bir model araştırılmıştır. Gerçekleştirilen çalışma kapsamında proje hedeflerini başarı ile tamamlayan ve tamamlamayan projelerin özellikleri incelenmiş ve bir projenin başarısına etki eden faktörler belirlenmiştir. Bir projenin başarı ile tamamlanması projenin sözleşmesinde belirtilen kapsam, süre, bütçe ve işgücü hedeflerinin başarılması anlamına gelmektedir. Başarılı projelerin geçmiş süreç performans verileri toplanarak süreç kontrol kartları kullanılarak süreç performans dayanakları oluşturulmuştur. Yeni başlayan projenin belirli aşamalarında alınan süreç ölçümleri süreç kontrol kartları yardımı ile değerlendirilmiş, sürecin kontrol altında olup olmadığı kararı verilmiştir. Ayrıca süreç metriklerine Analitik Hiyerarşi Süreci ile ağırlık verilmiştir. Ölçülen süreç metrik değeri ile o metriğe ait ağırlık katsayısının çarpılmasının ardından normalizasyon işlemi yapılarak proje performans başarı puanını hesaplanmaktadır.

Anahtar Kelimeler: Proje Yönetimi, Proje Performans Değerlendirme, İstatistiksel Süreç Kontrolü

Developing a Model for Measuring Project Performance with Software Project Life Cycle Process Metrics and Calculating Project Success Score

Özgür Gün[1], Pınar Yıldız Kumru[2]

[1,2] Industrial Engineering Department, Kocaeli University, Kocaeli, Turkey
ozgur.gun@tubitak.gov.tr, pinarki@kocaeli.edu.tr

Abstract. In this thesis, a model that evaluates the performance of system and software projects by using project life cycle process metrics and calculates a

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

project performance achievement score is investigated. Within the scope of the study, the characteristics of the projects that have successfully completed and not completed the project objectives have been examined and the factors affecting the success of a project have been determined. Successful completion of a project means achieving the scope, duration, budget and workforce objectives set out in the project contract. Past performance data of successful projects were collected and process performance bases were established using process control cards. Process metrics measured at certain stages of the newly started project were evaluated with the help of process control cards and it was decided whether the process was under control or not. In addition, process measurement metrics were weighted with Analytic Hierarchy Process. After multiplying the measured process metric value and the weight coefficient of that metric, the project performance achievement score is calculated by performing normalization process.

Keywords: Project Management, Project Performance Evaluation, Statistical Process Control

1 Giriş

Ülkemizde enerji, sağlık, havacılık, uzay, otomotiv, raylı sistemler, bilişim, savunma, bilgi güvenliği ve kamu dijital dönüşüm sektörleri gibi öncelikli sektörlerde teknolojik ürün yatırımlarının ve kümelenme çalışmalarının desteklenmesi yönünde politikalar oluşturulmuştur [1],[2],[3]. Bu alanlarda başlatılan proje sayısında büyük bir artış olmuştur. Ülkemiz için önem arz eden bu projelerin başarıyla sonuçlandırılması da büyük önem arz etmektedir. Fon sağlayıcı kurumların başlattıkları projelerin izlenmesi konusunda takip ettikleri yöntemler mevcuttur. Belirli dönemlerde bir hakem heyeti ile projelerde denetim yapılmaktadır. Bu denetimler sonucunda projelerdeki riskli konular belirlenmektedir. Fakat projelerin belirli aşamalarında sayısal bir değerlendirme modeline göre projenin performans başarı puanını gösteren bir sistem bulunmamaktadır.

Bu tez çalışmasına girdi teşkil eden veriler yazılım ve sistem projeleri geliştiren bir Ar-Ge kurumundaki projelerden toplanmıştır. Bahse konu Ar-Ge kurumu savunma ve sivil sektörde proje geliştiren ve 400'ü aşkın çalışanı bulunan bir kurumdur. Kurumdaki toplam 25 projeden veri toplanmıştır.

2 İlişkili Çalışmalar

Literatürde, proje başarı faktörleri ile ilgili çalışmalar mevcuttur. Bu bölümde bu çalışmalara kısaca değinilecektir.

Akyol [4] doktora tez çalışmasında farklı ölçek ve özelliklerdeki yazılım organizasyonlarında proje başarısını etkileyen parametreleri belirtmiştir. Sağlıklı verilerin elde edilmesinin çalışma kapsamında önemli kısıtlardan biri olduğunu ifade etmiştir.

Ayyıldız [5] yeni bir yazılım ölçüt kümesi oluşturma, oluşturulan yazılım ölçüt kümesi için veri toplama ve yeni veri kümesi ile yapay sinir ağı kullanılarak yazılım

maliyet tahmin modeli oluşturma çalışması yapmıştır. Veri toplamakta ciddi problemlerin olduğunu çünkü bu verilerin birçok şirket için gizli olduğunu ve paylaşılmasının şirketin rekabet ortamı için dezavantaj oluşturduğunu belirtmiştir.

Çatal [6] yazılım süreç metriklerini kullanarak yapay bağışıklık sistemini kullanarak yazılım kusur kestirimi modelini ortaya çıkarmıştır.

Radujković vd. [7] proje yönetimi başarısını etkileyen faktörlerini proje yönetici ve çalışanları yetkinliği, organizasyon yapısı ve kültürü ve kullanılan metodoloji, araç ve teknik olarak tanımlar.

Sanchez vd. [8] bilgi sistemleri geliştirme projelerini inceleyerek proje başarısının temel yönlerini çok düzeyli bir bakış açısıyla ele alarak akademik literatüre katkıda bulunmuşlardır. Çalışmalarında proje büyüklüğü, süre, erteleme ve proje yöneticisi yetkinliği, takım büyüklüğü ve ekip dağılımı faktörlerinin projeye maliyet ve zaman açısından etkilerini incelemişlerdir.

Mir vd. [9] proje yönetimi (PY) ile ilgili literatürün, PY süreçlerinde, araç ve sistemlerde kaydedilen ilerlemeye rağmen, proje başarısının önemli ölçüde iyileşmediğini iddia ederek PY performansı ile proje başarısı arasındaki ilişkiyi test eden bir araştırma yapmışlardır.

Todorović vd. [10] kritik başarı faktörlerini ve performans göstergelerini temel alarak proje başarı analiz çerçevesi sunmuşlardır.

Verner vd. [11] proje başarısına etki eden faktörleri araştırmışlardır.

Garousi vd. [12] yazılım proje başarısı ile kritik başarı faktörleri arasındaki korelasyonu araştırmışlardır.

3 Süreç Metriklerinin Belirlenmesi

Süreç metriklerinin belirlenmesi, bir Ar-Ge kurumundaki süreç iyileştirme ekip çalışanlarıyla yapılan çalıştaylar sonucunda belirlenmiştir. Metrikleri belirlenirken var olan süreç metriklerinin yanında benzer kurumlardaki süreç metrikleri de incelenmiştir. Belirlenen metriklerin özeti Tablo 1'de verilmiştir. Tablo 1'de kullanılan kısaltmalar sırasıyla aşağıdaki gibidir.

MTR: Müşteri Tarafından Raporlanan
AKS: Açık Kalma Süresi
AKO: Açık Kalma Oranı
HY: Hata Yoğunluğu

Tablo 1. Süreç Metrikleri Özeti

Süreç Adı	Metrik Adı						
Proje Yönetimi	Kilometretaşı Sapması	Efor Sapması	Gelir/Gider İndeksi				
Gereksinim Mühendisliği	Proje Başına Ortalama Gereksinim Değişikliği Sayısı						
Geliştirme	MTR Hata AKS	MTR İyileştirme AKS	MTR Gereksinim AKS	MTR Hata AKO	MTR İyileştirme AKO	MTR Gereksinim AKO	H Y
Konfigürasyon Yönetimi	Kon. Den. Pl. Sapma	Konf. Den. Ort. Uygunsuzluk Say.	Konf. Uygunsuzluk AKS	Konf. Uygunsuzluk AKO			
Risk Yönetimi	Risk Gözden Geçirme Oranı						
Kalite Güvence	Uygunsuzluk AKS	Denetim Pl. Sapma					
Doğrulama ve Geçerleme	Gözden Geçirme Haz. Hız	Bulgu Yoğ.	Hazırlık Verimliliği	Hata Yakalama Yüzdesi	Hata Tespit Verimliliği		

Tablo 1'de gösterilen süreç metriklerinin hem proje bazında hem de kurum bazında veri toplama ve analiz yöntemleri tanımlanmıştır. Süreç performans dayanaklarını belirlemek amacıyla oluşturulan süreç kontrol kartları yardımıyla, veri toplanan toplam 25 projeden alınan verilerin değerlendirilmesi sonucunda her bir süreç bazında kurumsal süreç performansı ortaya çıkmıştır. Proje performans başarı puanı hesaplanırken, projenin süreç performansı kurum süreç performansı ile karşılaştırılmaktadır. Bu nedenle hem proje bazında hem de kurumsal düzeyde süreç metrik analiz yöntemleri tanımlanmıştır.

4 Proje Geliştirme Araç Altyapısı

Verilerin toplandığı Ar-Ge kurumunda proje geliştirme araçları Tablo 2'de gösterilmiştir.

Tablo 2. Proje Geliştirme Araçları

Proje Planlama ve Yönetim	Atlassian JIRA
	Atlassian Confluence
	IBM Rational Team Concert
	MS Project
Gereksinim Yönetimi	IBM DOORS
	IBM DOORS Next Generation
	Atlassian JIRA
Analiz / Tasarım	Enterprise Architect

	IBM Rational Rhapsody
	IBM Rational Software Architect
	Altium Designer
Kodlama	Eclipse, Visual Studio
Birim Test	Junit
	Google Code Pro
	Quick Test Professional
Paket Yönetimi(Build) Ve Tümleştirme	Atlassian Bamboo
	Jenkins
	Ant
	IBM Rational Team Concert
Test Yönetim	Rational Quality Manager
Konfigürasyon Yönetim	IBM Rational Team Concert
	SVN (Subversion)
	GIT
İş Yönetim	Atlassian JIRA
	IBM Rational Team Concert
Gözden Geçirme	SmartBear Collaborator
Yazılım Kalitesi Ölçme Aracı	Understand
	Sonar Source

Tablo 2'de belirtilen proje araçlarında işletilen süreç iş akışları ve bu iş akışlarının koştuğu yazılım araçları Tablo 3'te belirtilmiştir.

Tablo 3. Proje Süreç İş Akışları

Sıra	İş Akış Adı	İlgili Araç
1	Proje iş aktiviteleri	IBM Rational Team Concert, Atlassian JIRA
2	Değişiklik istekleri	IBM Rational Team Concert, Atlassian JIRA
3	Hata bildirimleri	IBM Rational Team Concert, Atlassian JIRA
4	Risk faaliyetleri	IBM Rational Team Concert, Atlassian JIRA
5	Toplantılar	IBM Rational Team Concert, Atlassian JIRA
6	Süreç denetimleri	IBM Rational Team Concert, Atlassian JIRA
7	İş ürünü konfigürasyon denetimleri	IBM Rational Team Concert, Atlassian JIRA
8	Gözden geçirmeler	SmartBear Collaborator

9	Düzeltici faaliyetler	IBM Rational Team Concert, Atlassian JIRA
10	Öğrenilen dersler	IBM Rational Team Concert, Atlassian JIRA

5 Süreç Performans Dayanaklarının Oluşturulması

Tablo 1'de belirtilen her bir süreç metriği için süreç performans dayanağı oluşturulmuştur. Bu bölümde örnek olarak gözden geçirme sürecine bağlı gözden geçirme hazırlık hızı metriği için süreç performans dayağının nasıl oluşturulduğu belirtilmiştir. Gözden geçirme hazırlık hızı süreç kontrol kartının oluşturulabilmesi için kurumdaki toplam 25 yazılım projesinden gözden geçirme hazırlık hızı bilgileri toplanmıştır. Proje bazında ve kurum bazında gözden geçirme hazırlık hızının nasıl hesaplandığı aşağıda belirtilmiştir.

Ölçüm Amacı:

Proje Düzeyinde: Projede gerçekleştirilen gözden geçirme faaliyetleri için yeterli süre ayrılıp ayrılmadığını, gözden geçirme bulgularının yoğunluğunu ve birim gözden geçirme zamanında tespit edilen bulguların sayısını ölçmek.

Kurum Düzeyinde: Kurumda yürütülen projelerde gerçekleştirilen gözden geçirme faaliyetleri için ayrılan sürelerin, gözden geçirme bulgularının yoğunluğunun ve birim gözden geçirme zamanında tespit edilen bulguların sayısının merkezî eğilimini tespit etmek

Proje Bazında Gözden Geçirme Hazırlık Hızı = Hazırlık saati başına düşen büyüklük (sayfa sayısı ya da LLOC-logical line of code)

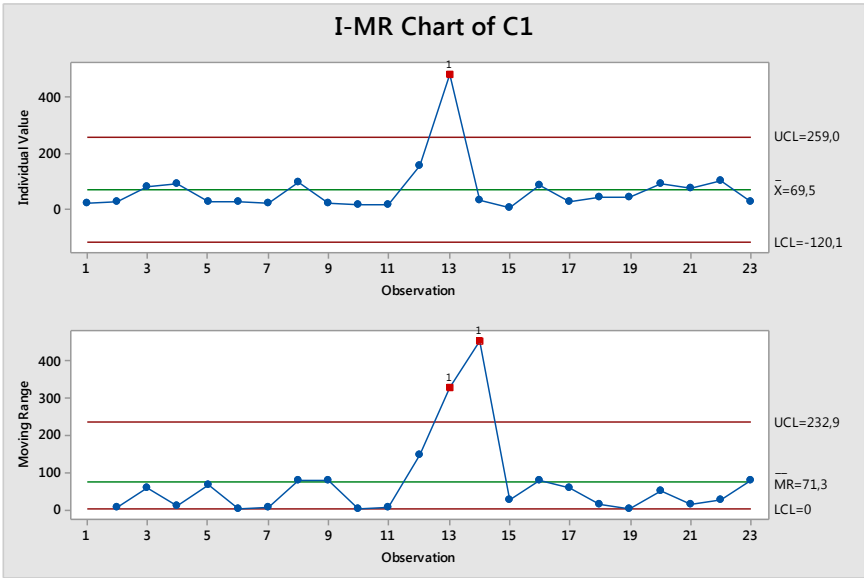
Enstitü Bazında Gözden Geçirme Hazırlık Hızı = Medyan (Proje Bazında Gözden Geçirme Hazırlık Hızı)

Kurumdaki projelerden 24 ay boyunca alınan veriler Tablo 3'te gösterilmiştir. Her ay değeri ilgili ayda tüm projelerde gerçekleştirilen gözden geçirme hazırlık hızı değerlerinin medyan ortalamalarının alınması ile hesaplanmıştır. Genellikle yazılım dünyasında kontrol kart tipi olarak U chart ya da I-MR kontrol kartları kullanılmaktadır [13]. Eğer verilerin dağılımı poisson ya da binominal dağılıma uygun ise U chart kontrol kartı kullanılabilir. Verilerin dağılımı konusunda kesin fikir yok ise I-MR kartı kullanılır [13]. Süreç kontrol kartı oluşturmak için Minitab aracında I-MR kontrol kartı grafiği kullanılmıştır.

Tablo 4. Gözden Geçirme Hazırlık Hızı Ölçümleri

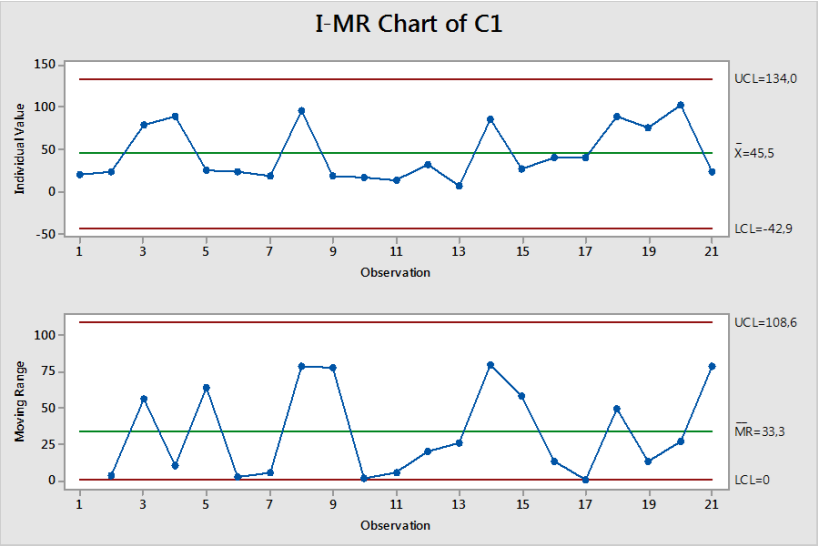
Sıra	Aylar	Gözden Geçirme Hazırlık Hızı (sayfa/saat)
1	2017-Ocak	21
2	2017-Şubat	24
3	2017-Mart	80
4	2017-Nisan	90
5	2017-Mayıs	26
6	2017-Haziran	24
7	2017-Temmuz	19
8	2017-Ağustos	97
9	2017-Eylül	19
10	2017-Ekim	18
11	2017-Kasım	13
12	2017-Aralık	158
13	2018-Ocak	485
14	2018-Şubat	33
15	2018-Mart	7
16	2018-Nisan	1507
17	2018-Mayıs	86
18	2018-Haziran	28
19	2018-Temmuz	41
20	2018-Ağustos	40
21	2018-Eylül	89
22	2018-Ekim	76
23	2018-Kasım	103
24	2018-Aralık	25

Tablo 4'deki veriler ile oluşturulan kontrol kartı Şekil 1'de gösterilmiştir.



Şekil 1. Gözden Geçirme Hızı Kontrol Kartı-Birinci İterasyon Sonucu

Şekil 1'de kontrol limitleri dışına çıkan noktaların analizleri yapıldıktan sonra kalan veriler ile tekrar kontrol kartı oluşturulmuştur. Üçüncü iterasyon sonucunda kalan veriler stabil bir süreç kontrol kartı grafiği oluşturmuştur.



Şekil 2. Gözden Geçirme Hızı Kontrol Kartı- Üçüncü İterasyon Sonucu

Şekil 2'de sürecin süreç stabil hale geldiği görülmüştür.

6 Süreç Metriklerinin Analitik Hiyerarşi Yöntemi (AHP) ile Ağırlıklandırılması

Proje yönetimi konusunda tecrübeli kurum çalışanları ile yapılandırılmış AHP çalışması yapılmıştır. Kurum çalışanları ile görüşmeler yapılarak süreç metrikleri için daha önce hazırlanan ikili karşılaştırma matrislerini doldurmaları istenmiştir. AHP çalışmasında kullanılan bağıl önem cetveli Tablo 5'de verilmiştir.

Tablo 5. Bağıl Önem Cetveli

Ağırlık	Tanım	Açıklama
1	Eşit	Her iki alternatif eşit derecede öneme sahip
3	Orta	İlk alternatif ikinciden bir miktar daha önemli
5	Kuvvetli	İlk alternatifin ikinciye göre önemi kayda değer
7	Çok kuvvetli	İlk alternatifin ikinciye göre önemi çok fazla
9	Çok önemli	İlk alternatifin ikinciye göre önemi son derece yüksek
2,4,6,8		Ara değerler

AHP sonuçları Tablo 6'da gösterilmiştir.

Tablo 6. Proje Yaşam Döngüsü Süreç Metrik Ağırlık Sonuçları

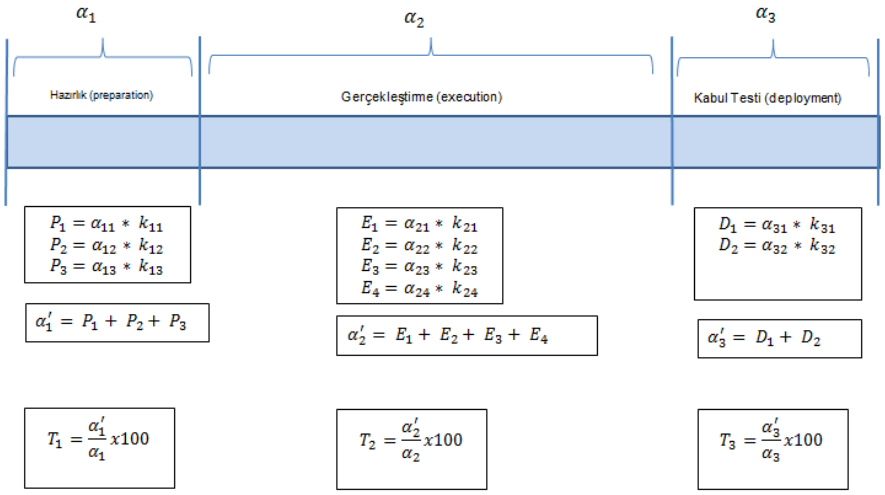
Metrik Adı	Ağırlık			**Metrik Adı**	Ağırlık
Proje Yönetimi	0,135			**Kalite**	0,078
Kilometretaşı sapması	0,052			Proje Uygunsuzluk AKS	0,048
Efor sapması	0,033			Proje Denetim Planlamasından Sapma	0,029
Proje Gelir/Gider İdeksi	0,050				
Geliştirme	0,195			**Doğrulama ve Geçerleme**	0,195
MTR Hata AKS	0,036			Gözden Geçirme Hazırlık Hızı	0,014
MTR İyileştirme AKS	0,011			Bulgu Yoğunluğu	0,051
MTR Gerek. Değ. AKS	0,030			Hazırlık Verimliliği	0,021
MTR Hata AKO	0,033			Hata Yakalama Yüzdesi	0,049
MTR İyileştirme AKO	0,011			Hata Tespit Yüzdesi	0,060
MTR Gerek. Değ. AKO	0,029				
Hata Yoğunluğu	0,045			**Gereksinim Yönetimi**	0,21
Konfigürasyon Yönetimi	0,108			Proje Başına Düşen Ort.Ger. Değiş.Sayısı	0,21

Konfigürasyon Denetim Planlamasından Sapma	0,023			**Risk Yönetimi**	0,08
Konfigürasyon Den. Başına Düşen Ort. Hata	0,025			Risk Gözden Geçirme Oranı	0,08
Konfigürasyon Uygunsuzluklarının AKS	0,035				
Konfigürasyon Uygunsuzluklarının AKO	0,024			Toplam ağırlık	1

MTR: Müşteri tarafından raporlanan AKS: Açık kalma süresi AKO: Açık kalma oranı

7 Proje Performans Başarı Puanı Hesaplama Modeli

Şekil 3'te proje performans başarı puanı hesaplama modeli gösterilmiştir. Yapılan varsayımda hazırlık aşamasında üç, yürütme aşamasında dört ve kabul aşamasında iki olmak üzere toplam 9 süreç metriğinin ölçüldüğü varsayılmıştır.



Şekil 3. Proje performans başarı puanı hesaplama modeli

Şekil 3'te belirtilen proje aşamaları kurumdan kuruma farklılık gösterebilir. Her bir aşamada ölçülen süreç metrikleri de kurumların iş hedeflerine, ürün ve süreç ölçüm hedeflerine göre değişiklik gösterebilir.

k_{ij} AHP yöntemi ile hesaplanan metrik ağırlık katsayısıdır.

$\alpha_1, \alpha_2, \alpha_3$ beklenen başarı puanlarıdır.

$\alpha_{1n}, \alpha_{2n}, \alpha_{3n}$ ölçülen süreç metrik değeridir.

P_n, E_n, D_n ilgili aşamaya ait süreç metrik başarı puan değeridir.

T_1, T_2, T_3 ilgili aşamaya ait proje başarı puan değeridir.

Beklenen başarı puan değerlerinin hesaplanması: Proje aşamalarındaki beklenen başarı puanı, geçmiş proje süreç metrik ölçüm verilerinin analiz sonuçları ile konusunda uzman ve tecrübeli proje yöneticileri, iş paketi liderleri ve tecrübeli proje çalışanları ile yapılan mülakat ve anketlerden gelen bilgiler ile belirlenmiştir.

8 Sonuç

Bu tez çalışmasında projelerden toplanacak süreç metrikleri tanımlarının detaylandırılması (metriklerin amacı, toplanma ve analiz yöntemleri) yapılmış ve ölçümlerin saklanacağı veri tabanı tasarımı gerçekleştirilmiştir. Sonrasında projenin planlama, analiz, gerçekleştirme ve test aşama sonlarında proje süreç metriklerini otomatik olarak toplayıp analiz eden ve proje başarı puanını hesaplayan bir yazılım geliştirilmiştir. Geliştirilen yazılımın detay tasarımı da dokümante edilmiştir.

Yazılım java dilinde geliştirilmiştir ve web tabanlı çalışacak şekilde tasarlanmıştır. Yazılımda yeni metrik tanımlayabilme, verilerin toplanacağı proje destek araçlarını artırma ya da değiştirme, veri toplanan projeleri ekleme/çıkarma gibi işlemler yapılabilmektedir. Geliştirilen yazılım proje performans başarı puanı hesaplama modelini kullanarak projenin istenilen aşamalarında proje başarı puanını hesaplamaktadır. Yazılım istatistiksel süreç kontrolü altında olmayan süreçleri ve süreçle ilgili verileri de kullanıcıların görmesine izin vermektedir.

Kaynakça

1. T.C. Kalkınma Bakanlığı Onuncu Kalkınma Planı, Madde 681.
2. Orta Vadeli Program 2016-2018, Madde 180.
3. Bilişim Teknoloji ve Sanayi Bakanlığı Türkiye Yazılım Sektörü Stratejisi ve Eylem Planı 2017-2019.
4. Akyol, M. M.: Yazılım süreçlerini etkileyen faktörlerin belirlenmesine ilişkin bir ölçüm ve iyileştirme modeli ve uygulaması, İstanbul (2013).
5. Ayyıldız, M.: Yazılım projeleri ölçüm sonuçları veri tabanının oluşturulması ve yeni yazılım projelerinin maliyet tahmininde kullanımı, İstanbul (2007).
6. Çatal, Ç.: Yazılım kusur kestirimi probleminde yapay bağışıklık sistemlerinin uygulanması. İstanbul (2008).
7. Radujković, M., Sjekavica, M.: Project Management Success Factors. In: Creative Construction Conference, Procedia Engineering 196 (2017), pp. 607 – 615. Elsevier, Croatia (2017).
8. Sanchez, O.P., Terlizzi, M.A., Moraes, H.: Cost and time project management success factors for information systems development projects. In: International Journal of Project Management 35, pp. 1608–1626. (2017).
9. Mir, F.A., Pinnington, A.H.: Exploring the value of project management: linking project management performance and project success. In: International Journal of Project Management 32, pp. 202–217. (2014).

10. Todorović, M.J., Petrović, D.Č., Mihić, M.M., Obradović, V.L., Bushuyev, S.D.: Project success analysis framework: a knowledge-based approach in project management. In: International Journal of Project Management 33, pp. 772–783. (2015).
11. Verner, J., Kitchenham, B., Cerpa, N.: Estimating Project Outcomes. In: Conference Paper, ICSSEA 2007-11, (2007).
12. Garousi, V., Tarhan, A., Pfahl, D., Coşkunçay, A., Demirörs, O.: Correlation of critical success factors with success of software projects: an empirical investigation. Software Quality Journal 27, pp. 429–493. (2019).
13. Florac, W. A., Park, R. E., Carleton, A. D.: Practical software measurement: measuring for Process Management and Improvement. Guidebook, CMU/SEI-97-HB-003, (1997).

Test Olgunluk Modelinin (TMMI) Agile ve Test Süreçleri ile Bütünleştirilmesi

Ahmet Unudulmaz[1] , Özgür Cingiz[2], Oya Kalıpsız[3]

[1,3] Yıldız Teknik Üniversitesi, İstanbul, Türkiye
ahmetunudulmaz@gmail.com, okalipsiz@gmail.com
[2] Bursa Teknik Üniversitesi, İstanbul, Türkiye
mustafa.cingiz@btu.edu.tr

Özet. Test Olgunluk Modeli (TMMI) test süreçlerini iyileştirmek için kullanılan belirli seviyeleri olan bir mimari modeldir. Bu yaklaşımın test süreçlerine entegre edilmesiyle birlikte proje ve ürün kalitesi üzerinde olumlu etkileri olmakta, test eforunu ise düşürmektedir. Bu pozitif etkilerinin yanında TMMI uygulamak dokümantasyon açısından ağır bir yük getirmektedir. TMMI ve Scrum pratikleri bir arada kullanılarak hem test kalitesi arttırılabilir hem de mevcut olan bu doküman maliyeti azaltılabilir. Bu bildiride agile yazılım geliştirme metodolojisi ve test pratiklerinin TMMI Seviye – 2 ile entegrasyonu ele alınmıştır. TMMI Seviye – 2 alt adımları olan test politikası, test stratejisi, test planlama, test izleme ve kontrol etme, test tasarım ve test çalıştırma adımlarının hem agile pratikleri hem de bazı test pratikleri ile (Risk bazlı test teknikleri, Kullanıcı Senaryo test teknikleri) birlikte kullanılması üzerine çalışma yapılacaktır. TMMI kapsam yüzde oranları da proje sonunda belirtilerek projelere nasıl uyarlanacağından bahsedilecektir.

Anahtar Kelimeler: TMMI, Agile ve TMMI, Test Politikası, Test Stratejisi, Test Planlama, Test Tasarım.

Test Maturity Model Integration (TMMI) Level Two Implementation with Agile and Software Test Process

Ahmet Unudulmaz[1] , Özgür Cingiz[2], Oya Kalıpsız[3]

[1,3] Yıldız Teknik Üniversitesi, İstanbul, Türkiye
ahmetunudulmaz@gmail.com, okalipsiz@gmail.com
[2] Bursa Teknik Üniversitesi, İstanbul, Türkiye
ozgurcingiz@gmail.com

Abstract. TMMI has a staged architecture and detailed model for process improvement. Appliying this approach has a positive impact on project and product

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

quality and test effort. Besides positive impact, TMMI contains heavy documentation. Appliying scrum practice with TMMI can reduce the documentation and improve test process quality by applying both of them. In this paper agile and testing practices are mapped with TMMI Level – 2 Managed step. Test Policy and Strategy, Test Plannig, Test Monitoring and Control, Test Design and Execution and Test Environment areas are discussed and matched with not only scrum practice but also some test practice (Risk Based Testing, Use Case Testing). In the end TMMI coverage percentage will be given and Improvements areas will be defined. This model also can be adapted for every project which is using agile and scrum practice.

Keywords: TMMI, Managed, Agile with TMMI, Test Policy, Test Strategy, Test Plan, Test Design.

1 Giriş

Çevik süreç manifestosunda da yer aldığı üzere bu metodolojinin temelinde süreçler ve araçlardan ziyade bireyler ve etkileşimlere, kapsamlı dokümantasyondan ziyade çalışan yazılıma, sözleşme pazarlıklarından ziyade müşteri ile iş birliği ön plana çıkmıştır [1,2]. Çevik süreç testleri gereksinimlerin dinamik olarak değişebileceği, müşterinin de yazılım geliştirme ve test süreçlerine dahil olabildiği iteratif yazılım geliştirme metodu içinde yer alan süreçlerdir. Test fazı projenin başlangıç aşamasından bitiş anına kadar ele alınması gereken ve yazılımcı ile testçinin arasındaki entegrasyonu sağlayan süreçleri kapsamalıdır. Kalite bağlamında bakıldığında ise test yükü ve test aktiviteleri projeler için anahtar alanları oluşturmaktadır. Fakat günümüzde çevik süreç konseptleri kullanan projelerde genel olarak test süreçleri ihmal edilmekte, testler ve sonuçları takip edilmemekte ve test anlamında herhangi bir süreç geliştirme modeli veya test konsepti kullanılmakta veya daha az kullanılmaktadır. Çevik süreçler içinde bu eksiği kapatmak adına Test Olgunluk Modelinin (TMMI) uyarlanması amaçlanmıştır. Projelere TMMI uyarlandığında proje ve ürün kaliteleri üzerinde pozitif etki yarattığı, test süreçlerinin iyileştiği ve test yükü ve eforunun azaldığı bilinmektedir [3,4]. Bu faydalarının yanında TMMI ile birlikte ek olarak dokümantasyon ve limitasyon yükü de gelmektedir. Çevik süreçler ile birlikte kullanıldığında ise bu dokümantasyon yükünden kurtulunacağını ve süreçlerdeki eksik test yapısının da bu şekilde kapatılabilineceği düşünüldüğü için bu çalışma yapılmıştır.

Test olgunluk modelinin (TMMI) çevik süreçlerle entegrasyonu için ilk aşama olarak ikinci seviye ele alınmıştır. Seviye iki altındaki adımlara bakıldığında test politikası, test stratejisi, test planlama, test izleme ve kontrol etme, test tasarım ve çalıştırma ve test ortamı konuları bu çalışma kapsamında ele alınacak olup detayları ve çevik süreçlerle olan entegrasyonu anlatılacaktır. Uygulanabilir olan alanlar, yarı uygulanabilir alanlar belirtilecek, hiç uyarlanamayan alanlar özellikle belirtilerek çevik süreçlerle TMMI entegrasyonunun yapılıp yapılamayacağı üzerinde tartışılacaktır. TMMI seviye iki ve çevik süreç ve scrum pratikleri tartışılırken risk bazlı test teknikleri ve kullanıcı senaryo test teknikleri de gene bu çalışma kapsamında ele alınacak olup ilişkilendirme sürecinde bu kavramalara da yer verilecektir.

2 İlgili Çalışmalar

Malardalen üniversitesinde yazılım test süreç iyileştirme modelleri üzerine bir literatür taraması yapmışlar. 18 farklı test iyileştirme süreci incelenmiştir. Bu tarama sonrasında ise 5 model detaylandırılmıştır: TMM, TMMI, TPI Next, Test Spice, TPI Next. Çalışmanın ikinci aşamasında ise TPI Next ve TMMi modellerinin birbirleriyle olan farkları üzerine odaklanmışlardır. Buna göre bu iki model üzerindeki temel farklar ise model temsilleri üzerinedir. TMMi aşamalı bir model sunarken TPI Next ise sürekli bir model sunmaktadır [3].

Hacettepe üniversitesinde yapılan çalışmada ise gene Malardalen üniversitesinde yapılan çalışmaya paralel olarak test süreç iyileştirme modelleri üzerinde bir literatür taraması yapılmıştır. Tarama genelinde test olgunluk modeli (TMA) ile test süreç iyileştirme (TPI) bazlı modeller karşılaştırılmıştır. Toplamda 58 farklı süreç iyileştirme modeli kullanılmıştır [4].

TMMI vakfı tarafından yayınlanan dokümanlara göre TMMI seviyeleri bazında hangi alt alanların çevik süreçler ile uyumlu olduğunu belirtilmiştir. Buna göre seviye iki bazında test politikası ve test stratejisi, test planlama, test izleme, test dizayn ve koşum ve test ortam adımlarının büyük oranda çevik süreçlere uyarlanabilir olduğunu bildirmişlerdir [5,6].

Helwan üniversitesinde yapılan çalışmada ise scrum pratikleri ile TMMI ikinci seviyesinin adreslenmesi ve projedeki performans ve kalite faktörleri üzerine bir çalışma yapmışlar. Bu çalışmaya göre bütün test alt alanları scrum pratikleriyle eşleşmiştir. Test konfigürasyon kısmında ise %90 oranında eşleşmeyen veya yarı uygulanabilir alan bulmuşlardır [7].

3 Metodoloji

Bildiri kapsamında TMMI Seviye 2 ile çevik süreç ve test süreçlerinin entegrasyonu amaçlanmaktadır. Bütün alt alanlar ayrı ayrı ele alınarak çevik süreçlerle eşleşme durumları ele alınacaktır [6,7]. Çevik süreçlerle beraber risk bazlı test teknikleri ve kullanıcı senaryo test tekniklerinin de TMMI ile entegrasyonuna bakılacak ve değerlendirmeler buna göre yapılacaktır. Tablo 1'de de belirtildiği üzere eğer TMMI pratiği çevik süreç ve test süreç pratiği ile gerçeklenebiliyorsa "Evet uygulanır (E)", bazı alanlar uygulanır bazı alanlar uygulanamıyorsa yani çevik süreçlere tam anlamıyla uygun değil ama bazı pratiklerle gerçeklenebiliyorsa "Yarı uygulanır (Y)", eğer hiçbir pratikle eşleştirilemez ise "Hayır uygulanamaz (H)" seviyeleri ile eşleştirilecektir.

Tablo 1. Eşleştirme Kriterleri

Açıklama	Kriter
Pratik uygulanabilir	Evet uygulanır (E)
Pratik yarı uygulanabilir	Yarı Uygulanır (Y)
Pratik uygulanamaz	Hayır uygulanamaz (H)

Yapılacak çalışma kapsamında aşağıda belirtilen TMMI alt alanları ele alınacaktır [5]:

- Test politikası ve test stratejisi
  - Test politikası kurmak
  - Test stratejisi kurmak
  - Test performans göstergesi kurmak

- Test planlama
  - Risk değerlendirmesi yapmak
  - Test yaklaşımı geliştirmek
  - Test tahmininde bulunmak
  - Test planlaması geliştirmek
  - Test planına bağlı kalmak

- Test izleme ve kontrol
  - Plan kapsamında test ilerlemesinin izlenmesi
  - Plan ve beklentiler kapsamında ürün kalitesinin izlenmesi
  - Kapatmak için düzenleyici aktivitelerin yönetimi

- Test tasarım ve çalıştırma
  - Test tasarım teknikleri kullanarak test analiz ve tasarımın gerçeklemesi
  - Test implementasyonunun gerçeklemesi
  - Test koşumunun gerçeklemesi
  - Kapanış için test hatalarının yönetimi

- Test ortamı
  - Test ortamı gereksinimlerinin geliştirilmesi
  - Test ortamı implementasyonunu yapılması
  - Test ortamının kontrolü ve yönetilmesi

3.1 Test Politikası ve Test Stratejisi

TMMI ikinci seviyesinin ilk adımı olan test politikası ve test stratejisinin temel amacı test politikasının tanımlanması, organizasyon genelinde test stratejisinin ve test tiplerinin belirlenmesidir. Ayrıca test performansının ölçümü içinde test performans göstergeleri belirlenmeli ve geliştirilmelidir [6].
Tablo 2'de de görüleceği üzere test politikası, test stratejisi ve test performans göstergelerinin çevik süreçler ve test süreçleriyle eşleştirilmesi gösterilmiştir.

Tablo 2. Test Politikası ve Test Stratejisi

Adım	TMMI Pratikleri	Çevik Süreç ve Test Pratikleri	Eşleştirme
1.1 – Test Politikası	Test hedefi belirleme	Sprint planlama toplantıları	E
1.2 – Test Politikası	Test politikası belirleme	Projeye başlanmadan önce test politikasının belirlenmesi (Testin tanımı & Kalite seviyeleri)	E
1.3 – Test Politikası	Paydaşlarla test politikasının paylaşılması	Sürüm planlama toplantılarında bu bilginin paylaşılması gerekmekte. Fakat çevik süreç kullanan projelerde aynı süreç her zaman kullanılmamakta ve paydaşlar dahil edilmemektedir.	Y
2.1 – Test Stratejisi	Risk değerlendirmesinin gerçeklemesi	**Risk bazlı test yaklaşımı** kullanılarak bu adım uyarlanabilir. Sadece çevik süreç kullanımı yeterli olmamaktadır.	E
2.2 – Test Stratejisi	Test stratejisi belirleme	Projeye başlanmadan önce test stratejisinin belirlenmesi ve her sprintte gözden geçirilmesi (Sorumluluklar & Araçlar & Riskler)	E
2.3 – Test Stratejisi	Test stratejisinin paydaşlarla paylaşılması	Sürüm planlama toplantılarında bu bilginin paylaşılması gerekmekte. Fakat çevik süreç kullanan projelerde aynı süreç her zaman kullanılmamakta ve paydaşlar dahil edilmemektedir.	Y
3.1 – Test Perf. Göstergesi	Test performans göstergelerinin belirlenmesi	Sprint metrikleri tanımlanmalı, ilerlemenin ölçülmesi ve test kapsamının ölçülmesi metriklerinin kullanılması	E
3.2 – Test Perf. Göstergesi	Test performans göstergelerinin geliştirilmesi	Test performans araçlarının projeye entegre edilmesi ve her sprinte bu bilginin güncellenmesi gerekmektedir	E

3.2 Test Planlama

Test planlama aşamasının temel amacı belirlenmiş olan riskler ve test stratejisi özelinde bir test yaklaşımı kullanmak, iyi planlanmış yönetilebilir bir test planı sayesinde de test aktivitelerini gerçekleştirebilmektir. Buradaki anahtar alan ileriyi düşünerek test planı oluşturmaktır [6].

Tablo 3'de görüleceği üzere risk tahmini, test yaklaşımları, test süre belirleme, test planı ve plana taahhütler yer almaktadır. Bu alanların çevik süreçler ve risk bazlı test teknikleri ile ilişkilendirilmesinden bahsedilecektir.

Tablo 3. Test Planlama

Adım	TMMI Pratikleri	Çevik Süreç ve Test Pratikleri	Eşleştirme
1.1 – Risk Değerlendirme	Ürün risk kategori ve parametrelerinin belirlenmesi	**Risk bazlı test yaklaşımı** kullanılarak risk kategorileri ve test hedefleri belirlenebilir	E
1.2 – Risk Değerlendirme	Ürün risklerinin belirlenmesi	Ürün risklerini belirlemek için **risk bazlı test yaklaşımı** kullanılabilir (Nasıl test yapılacak, bağımlılıklar, etkilenen alt sistemler vb.)	E
1.3 – Risk Değerlendirme	Ürün risk analizi	**Risk bazlı test yaklaşımı** kullanılarak risk olasılıkları, etki değerleri kullanılarak ürün riskleri analiz edilebilir	E
2.1 – Test Yaklaşımı	Test edilecek özelliklerin belirlenmesi	**Risk bazlı test yaklaşımı** kullanılarak test öncelik numarası belirlenebilir. Yüksek önceliğe sahip riskler ilk olarak test edilir.	E
2.2 – Test Yaklaşımı	Test yaklaşımının belirlenmesi	Test yaklaşımı takım planlama & arıtma toplantılarında belirlenir (Her bir iş maddesi için test seviyesi ve test adımı belirlenir)	E
2.3 – Test Yaklaşımı	Girdi kriterinin belirlenmesi	Girdi kriterleri takım arıtma toplantılarında her bir iş maddesi için belirlenir	E
2.4 – Test Yaklaşımı	Çıkış kriterinin belirlenmesi	Tamamlanmanın tanımı (DoD) bu kriter için kullanılır	E
2.5 – Test Yaklaşımı	Askıya alma ve yeniden başlama kriterlerinin belirlenmesi	Günlük standup toplantılarında problemli alanlar konuşulmakta fakat scrum içinde askıya alma ve yeniden başlama kriterleri bulunmamaktadır.	H
3.1 - Test Tahmini	Üst seviye kırılım yapısının kurulması	Asgari pazarlanabilir ürünlerin (MMF) tanımlanması ve ürün rezervlerine eklenmesi (PBI) sprint planlama ve arıtma toplantılarında ele alınır.	E
3.2 - Test Tahmini	Test yaşam döngüsünün belirlenmesi	Test yaşam döngüsü proje yaşam döngüsü ile ilişkilendirilir.	E
3.3 – Test Tahmini	Test eforu ve maliyetlerin kestirimi	Kullanıcı senaryoları puanlama ve t-shirt boyutu hesaplama gibi yöntem-	E

		ler kullanılarak sprint planlama toplantılarında test eforu ve maliyetleri belirlenir.	
4.1 - Test Planlama	Test zamanlamasının belirlenmesi	Sprint planlama aktivitesinde test eforu tahminleme üzerinden zamanlama çıkarımı yapılmaktadır.	E
4.2- Test Planlama	Test yapacak kişilerin planlanması	Sprint & takım planlama toplantılarında scrum ekiplerinde test yapacak kişiler ve harcayacakları efor belirlenmektedir.	E
4.3- Test Planlama	Paydaşların dahil olmasının planlanması	Paydaşların bütün scrum aktivitelerine dahil olmasının sağlanması gerekmektedir. Bunun için bütün projeler düşünüldüğünde yarı uygulanır olarak işaretlenmiştir.	Y
4.4 – Test Planlama	Proje risklerinin belirlenmesi	Risk bazlı test yaklaşımı kullanılarak proje paydaşlarının katıldığı toplantıda riskler belirlenir.	E
4.5- Test Planlama	Test planını belirlemek	Test planı sprint planlaması içinde ele alınmaktadır.	E
5.1 – Test Planlamasına Uyum	Test planını gözden geçirmek	Günlük scrum toplantılarında test planı gözden geçirilmekte ve gerekirse değiştirilmektedir.	E
5.2- Test Planlamasına Uyum	İş ve kaynak seviyelerinin uzlaştırılması	Günlük scrum toplantılarında bu konu konuşmaktadır. Fakat kaynak seviyeleri ile ilgili ayrı bir toplantı yoktur.	Y
5.3- Test Planlamasına Uyum	Test plan taahhütlerinin sağlanması	Scrum master işlerin taahüte uygun yapılıp yapılmadığını, DoD'a uygun olduğundan emin olmaktadır.	E

3.3 Test İzleme ve Kontrol

Test izleme ve kontrol aşamasında ise test ilerlemesinin ve ürün kalitesinin izlenebilmesi ve bu sürecin anlaşılabilmesi amaçlanmaktadır. Bu sayede ürün kalitesinde ve planlama aşamasındaki beklentiler karşılanabilir, düzenleyici aksiyonlar gene bu süreçte alınarak kalite istenen seviye getirilebilir [6].

Tablo 4'de görüleceği üzere test ilerlemesinin izlenmesi, ürün kalitesinin izlenmesi, düzenleyici önlemlerin alınma aşamaları çevik süreçler ve test yöntemleriyle ilişkilendirilmiştir.

Tablo 4. Test İzleme ve Kontrol

Adım	TMMI Pratikleri	Çevik Süreç ve Test Pratikleri	Eşleştirme
1.1 – Test ilerlemesinin izlenmesi	Test planlama parametrelerinin izlenmesi	Sprint tahtaları ve online çözümler kullanılarak scrum pratikleri ile takip edilmektedir.	E
1.2- Test ilerlemesinin izlenmesi	Sağlanan ve kullanılan test ortam kaynaklarının izlenmesi	Test ortam kaynakları otomatik araçlar kullanılarak izlenmektedir. İnsan kaynağı da sprint tahtası ile izlenmektedir.	E
1.3- Test ilerlemesinin izlenmesi	Test taahhütlerinin izlenmesi	Test taahhütleri, takım taahhütleri ile birlikte günlük scrum toplantılarında ve değerlendirme toplantılarında kontrol edilmektedir.	E
1.4- Test ilerlemesinin izlenmesi	Proje risklerinin izlenmesi	Proje riskleri test otomasyon yaklaşımı sayesinde belirli periyotlarla kontrol edilmektedir.	E
1.5- Test ilerlemesinin izlenmesi	Paydaşlarının katılımının monitör edilmesi	Paydaşların günlük scrum & planlama & inceleme ve arıtma toplantılarına katılımı herhangi bir scrum prensibi ile değil manuel olarak kontrol edilmektedir.	Y
1.6- Test ilerlemesinin izlenmesi	Test ilerlemesinin incelemesi	Günlük scrum ve gözden geçirme toplantıları ile kontrol edilmektedir.	E
1.7- Test ilerlemesinin izlenmesi	Test ilerlemesinin kilometre taşı bazında ilerlenmesi	Günlük scrum ve gözden geçirme toplantıları ile kontrol edilmektedir.	E
2.1 – Ürün kalitesini izleme	Girdi kriterinin kontrol edilmesi	Günlük scrum toplantıları ile kontrol edilmektedir.	E
2.2 – Ürün kalitesini izleme	Hataların izlenmesi	Scrum tahtaları ve hata takip sistemleri ile izlenmektedir. Günlük scrum toplantıları ile de kontrolü sağlanmaktadır.	E
2.3 – Ürün kalitesini izleme	Ürün risklerinin izlenmesi	Ürün riskleri otomatize edilen senaryolar ile belirli periyotlarla kontrol edilmektedir.	E
2.4 – Ürün kalitesini izleme	Çıkış kriterinin izlenmesi	DoD'a uyum ile çıkış kriterleri izlenmektedir fakat scrum pratikleri içinde kontrol çok sağlıklı yapılamamaktadır.	Y
2.5 – Ürün kalitesini izleme	Askıya alma ve devam etme kriterlerinin izlenmesi	Günlük standup toplantılarında problemli alanlar konuşulmakta fakat	H

		scrum içinde askıya alma ve yeniden başlama kriterleri bulunmamaktadır.	
2.6 – Ürün kalitesini izleme	Ürün kalitesinin incelenmesi	Scrum içinde sprint gözden geçirmeleri yapılmaktadır fakat ürün kalitesi için ayrı bir toplantı yapılmamaktadır.	H
2.7 – Ürün kalitesini izleme	Ürün kalitesinin kilometre taşı bazında incelenmesi	Scrum içinde ürün kalite kilometre taşı bazında inceleme yapılmamaktadır	H
3.1 – Kapama aşaması için düzenleyici aktiviteler	Problem analizi	Sprint gözden geçirme ve günlük scrum toplantılarında problem analizi yapılmaktadır.	E
3.2- Kapama aşaması için düzenleyici aktiviteler	Düzeltici aksiyonların alınması	Scrum içinde geçmişe yönelik toplantılar ile düzeltici aksiyonlar alınmaktadır.	E
3.3- Kapama aşaması için düzenleyici aktiviteler	Düzeltici aksiyonların yönetimi	Scrum master öncülüğünde düzeltici aksiyonlar yönetilmektedir.	E

3.4 Test Tasarım ve Çalıştırma

Test tasarımı ve çalıştırma aşamasında ise test tasarım teknikleri kullanılarak, yapısal olarak testlerin çalıştırması sağlanıp test ilerleme kapasitesini arttırabilmektir. Bu sayede bu seviyede ilgili hatalar bulunabilir ve raporlanarak bir sonraki aşamaya geçilebilir [6].

Tablo 5'de görüleceği üzere test analiz ve tasarım, test implementasyonu, test koşumu ve hataların raporlanması aşamaları çevik süreçler ve test süreçleri ile ilişkilendirilmiştir.

Tablo 5. Test Tasarım ve Çalıştırma

Adım	TMMI Pratikleri	Çevik Süreç ve Test Pratikleri	Eşleştirme
1.1- Test Analiz ve Tasarım	Test durumlarının önceliklendirilmesi	Risk bazlı test tekniği kullanılarak testler önceliklendirilmektedir.	E
1.2- Test Analiz ve Tasarım	Test senaryolarının önceliklendirilmesi	Risk bazlı test tekniği kullanılarak testler önceliklendirilmektedir.	E
1.3- Test Analiz ve Tasarım	Gerekli test verilerinin belirlenmesi	Her proje için test verisi gerekli olmamaktadır.	Y
1.4- Test Analiz ve Tasarım	Gereksinimlerin yatay düzeyde takip edilebilirliği	Takip edilebilirlik matrisi kullanılabilir.	E

2.1- Test İmplementasyonu	Test prosedürlerinin geliştirilmesi ve önceliklendirilmesi	**Kullanıcı senaryo test tekniği** kullanılarak test prosedürleri ortaya çıkarılabilir	E
2.2- Test İmplementasyonu	Spesifik test verisinin yaratılması	Her proje için test verisi gerekli olmamaktadır.	Y
2.3- Test İmplementasyonu	Giriş test prosedürünün belirtilmesi	Kullanıcı senaryo test tekniği kullanılarak pozitif ve negatif testler kullanılır	E
2.4- Test İmplementasyonu	Test koşum zamanının geliştirilmesi	Sprint planlama ve takım planlama ile adreslenmiştir	E
3.1 – Test Çalıştırma	Giriş testinin yapılması	Birim ve bileşen testlerin otomatik koşturulması ile adreslenmektedir.	
3.2- Test Çalıştırma	Test senaryolarının koşumu	Testlerin farklı seviyelerde koşturulması ile adreslenmektedir.	E
3.3- Test Çalıştırma	Test bulgularının raporlanması	Hata takip sistemleri kullanılarak bulgular raporlanabilir.	E
3.4- Test Çalıştırma	Test çıktılarının yazılması	Test otomasyon araçları kullanılarak çıktılar otomatik olarak depolanmaktadır. Her test maddesi için anlamlı log bilgisi yazmak önemli olmaktadır.	E
4.1 – Kapama için test bulgularının yönetimi	Hataların dağılımının kontrol edilmesi	Hata takip sistemleri kontrol edilerek bunlar üzerinde modeller çıkarılmaktadır.	E
4.2- Kapama için test bulgularının yönetimi	Hataların düzeltilmesi için düzeltici aksiyonlar	Hataların düzeltilmesi için yeni iş planları yaratılarak scrum tahtası üzerinden kontrol edilmektedir.	E
4.3- Kapama için test bulgularının yönetimi	Test hatalarının durumunun takibi	Hataların durumunun kontrolü için hata takip sistemi ve scrum tahtaları kullanılmaktadır.	E

3.5 Test Ortamı

Test ortamının amacı ilgili test verilerini içererek testlerin düzgün bir şekilde ve tekrarlı bir şekilde çalıştırılabilmesini sağlamaktır. Burada sadece test teknikleri ve çevik süreç yeterli olmamaktadır [6].

Tablo 6'da test ortamının gerçeklemesi, test ortam implementasyonunun gerçeklenmesi, test ortamının kontrol ve yönetimi çevik süreçlerle adreslenmeye çalışılmıştır. Tablo 6'da birçok alan sadece çevik süreçler kullanılarak TMMI seviye iki ile direkt olarak adresleyememiştir. Burada yazılım operasyon yaklaşımlarını da sürece dahil etmek gerekmektedir. Bu çalışma kapsamında sadece çevik süreçler ve test süreçleri dahil edildiği için diğer alt adımlarda olduğu gibi yüksek bir kapsama çıkmamaktadır.

Tablo 6. Test Ortamı

Adım	TMMI Pratikleri	Çevik Süreç ve Test Pratikleri	Eşleştirme
1.1- Test ortam gereksinimlerinin geliştirilmesi	Test ortam gereksinimlerinin ortaya çıkarımı	Sprint planlama ve arıtma toplantıları ile gereksinimler ortaya çıkarılır	E
1.2- Test ortam gereksinimlerinin geliştirilmesi	Test ortam gereksinimlerinin geliştirilmesi	Sprint planlama ve arıtma toplantıları ile gereksinimler geliştirilir	E
1.3- Test ortam gereksinimlerinin geliştirilmesi	Test ortam gereksinimlerinin analiz edilmesi	Test mimarı veya test yöneticisi gereksinimlerin analizinden sorumludur.	H
2.1 – Test ortam implementasyonunun gerçeklemesi	Test ortamlarının implementasyonu	Test ortamının implementasyonundan sprint günleri dahilinde takımlardaki testçiler sorumludur.	E
2.2- Test ortam implementasyonunun gerçeklenmesi	Kapsamlı test verisinin yaratılması	Bütün projeler için test verisine ihtiyaç duyulmamaktadır.	Y
2.3- Test ortam implementasyonunun gerçeklenmesi	Giriş test prosedürlerinin test ortamına uyarlanması	Giriş test prosedürleri scrum pratikleri ile adreslenememektedir.	H
2.4- Test ortam implementasyonunun gerçeklenmesi	Test ortamı için giriş testlerinin gerçeklenmesi	Scrum pratikleri içinde test ortamı için giriş testleri adreslenememektedir.	H
3.1 – Test ortamının kontrolü ve yönetimi	Sistem yönetiminin gerçeklenmesi	Sistem yönetimi çevik süreçler ile adreslenememektedir. Yazılım operasyon yaklaşımları kullanılmalıdır.	H
3.2- Test ortamının kontrolü ve yönetimi	Test veri yönetiminin gerçeklenmesi	Bütün projeler için test verisine ihtiyaç duyulmamaktadır.	H
3.3- Test ortamının kontrolü ve yönetimi	Test ortamının kullanılabilir olduğunun gösterilmesi	Test otomasyon sistemleri bu madde için kullanılabilir.	E
3.4- Test ortamının kontrolü ve yönetimi	Test ortamını hatalarının raporlanması ve yönetimi	Hata takip ara yüzü ve scrum tahtaları bu madde için kullanılabilir.	E

4 Sonuçlar ve Öneriler

Yapılan çalışma kapsamında test olgunluk modeli (TMMI) seviye iki altında bulunan adımların çevik süreçler ile entegrasyonundan bahsedilmiştir.Toplamda 5 başlık altında

70 alt başlık incelenmiştir. Bunlardan 51 tanesi çevik süreçler, risk bazlı test yaklaşımı ve kullanıcı senaryo test yaklaşımı ile adreslenebilinmiştir. 9 tanesi ise herhangi bir scrum pratiği ile adreslenememiştir. 10 tanesi ise parçalı olarak scrum ve test pratikleri ile adreslenmiştir. Tek başına scrum bütün alt maddeleri adreslemek için yeterli gelememektedir. Örneğin test ortamı alt maddesinde de bahsedildiği üzere yazılım operasyon yaklaşımlarının da bu sürece dahil edilmesi gerekmektedir. TMMI kapsam yüzdesi %72,8 olacak şekilde belirlenmiştir.

Yapılan bu çalışma ileride yapılacak olan çalışmaların temelini oluşturmaktadır. Sonraki aşamalarda beş TMMI seviyesi çevik süreçler ve test süreçleriyle eşleştirilecek ve yeni bir model ortaya konacaktır. Test yaklaşımının ikinci plana itildiği süreçlerde bu yeni modeli kullanmak kalite ve test süreçleri anlamında da olumlu bir etki sağlayacaktır. Oluşturulacak bu yeni modelle birlikte proje ve ürün kalitesinin ölçümü hedeflenmektedir. Proje aşağıdaki metriklerle ölçülerek iyileştirilen alanlar TMMI öncesi ve sonrası ortaya konulacaktır [7]. Tahmin edilen ve gerçeklenen test eforu ve test maliyeti, tahmin edilen ve gerçeklenen test senaryo sayısı, önceliklerine göre bulunan hata sayıları, toplam hata sayıları, hata bulma oranı, hata yoğunluğu, meslektaş değerlendirme kapsamı, kod kapsamı, gereksinim kapsamı, haftalık koşturulan test senaryoları metriklerinin detaylı ölçümü gerçekleştirilecektir.

İleride yapılacak çalışmalarda yukarıda bahsedilenlere ek olarak kapsam yüzdesini daha da yukarılara çıkarmak hedeflenmektedir. Bunun için çevik süreç pratiklerinin yanında farklı test pratikleri, yazılım operasyon yaklaşımları ve farklı test süreç iyileştirme modellerinin de kullanılması amaçlanmaktadır.

Kaynakça

1. Ken S., Jeff S.: The Definitive Guide to Scrum The Rules of the Game, The Scrum Guide (2017).
2. Kent B., Mike B., Arie v., Alistair C., Ward C., Martin F., James G., Agile manifeto, https://agilemanifesto.org/iso/tr/manifesto.html
3. Wasif A., Snehal A., Kerstin G., Richard T.: Software test process improvement approaches A systematic literature review and an industrial case study, The Journal of Systems and-Software, pp.1-33, Elsevier (2016)
4. Vahid G., Michael F., Tuna H.: Software test maturity assessment and test process improvement A multivocal literature review, Information and Software Technology, pp.16-42, Elsevier (2017)
5. TMMI Foundation, Test Maturity Model Integration (TMMI) Guidelines for Test Process Improvement,https://tmmi.org/tm6/wp-content/uploads/2018/11/TMMi-Framework-R1-2.pdf
6. TMMI Foundation, TMMI in the Agile World, Version 1.3, https://www.tmmi.org/tm6/wp-content/uploads/2019/07/TMMi-in-the-Agile-world-V1.3.pdf
7. Ahmed B., Enas M., Mahmoud A.: Towards Agile Implementation of Test Maturity Model Integration (TMMI) Level 2 using Scrum Practices, Interantional Jouranl of Advanced Computer Scienes and Applications Vol. 6 (2015)
8. Vahid G., Michael F., Tuna H.: What We Know about Software Test Maturity and Test Process Improvement, IEEE Computer Society (2018)

Fourier Açılımı Yöntemiyle Mutant Seçimi

Savaş Takan ve Tolga Ayav

İzmir Yüksek Teknoloji Enstitüsü
Bilgisayar Mühendisliği Bölümü, Urla, İzmir.
{savastakan,tolgaayav}@iyte.edu.tr

Özet. Mutasyon analizi, devre ve yazılım testlerinde test kümelerinin etkinliğinin ölçülmesinde yaygın olarak kullanılan bir tekniktir. Buna göre, orijinal model belirli hata varsayımları altında sistematik olarak mutasyona uğratılır ve testlerin bu yaratılan mutantların ne kadarını öldürdüğü kontrol edilir. Mutasyon analizinin hesaplama yoğunluğu genellikle mutant sayısının çok fazla olması nedeniyle yüksektir. Mutasyon analizinde her bir mutantın hata kapsama oranı bakımında eşdeğer olduğu varsayılır, ancak durum her zaman böyle değildir. Bu çalışma Boolean işlevlerinin Fourier analizine dayanan bir mutant seçim yöntemi önermektedir. Fourier, işlevin çıkışı üzerinde en fazla etkisi olan geçişleri belirleyerek bu geçişlerle ilgili olan mutantların seçilmesine yardımcı olur. Çalışmada, Fourier dönüşümü kullanılarak işlevin çıkışı üzerinde en fazla etkiye sahip olduğu tespit edilen bölgeler, literatürde yer alan mutant analiz yöntemleri ile karşılaştırılmıştır.

Anahtar Kelimeler: Mutasyon Analizi, Sonlu durum makinesi, Fourier dönüşümü, W yöntemi

Mutant Selection Using Fourier Expansion

Savaş Takan and Tolga Ayav

İzmir Institute of Technology
Dept. of Computer Engineering, Urla, İzmir
{savastakan,tolgaayav}@iyte.edu.tr

Abstract. Mutation analysis is a widely used technique to evaluate the effectiveness of a test suite in both hardware and software testing. The original model is mutated systematically under certain fault assumptions and test cases are checked against the created mutants whether the test cases kill them or not. Mutation analysis is a computationally-intensive task, particularly in Finite State Machine (FSM) testing due to a possibly huge amount of mutants. Mutation analysis usually assumes that each mutant is identical in terms of its fault detection effectiveness, which is not always the case. This study proposes a mutant selection method based on Fourier analysis of Boolean functions. Fourier helps to determine the most effective transitions on the output so that the mutants related to those transitions can be selected. The regions that are found to be the most influential are verified with the existing mutation analysis

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

methods.

Keywords: Mutation Analysis, Finite State Machine, Fourier transformation, W method

1 Giriş

Yazılım testlerinde öne çıkan araştırma konularından biri de test kümesi üretilmesinde kullanılan biçimsel modellerdir [6,9]. Bu modeller (FSM, Petri Net ve UML) ile oluşturulan testlerin etkinliğini ölçmek için genellikle mutasyon analizinden yararlanılmaktadır [14]. Mutasyon analizi kullanışlı bir teknik olmakla birlikte, çok fazla sayıda mutant üretilmesi nedeniyle zaman ve alan karmaşıklığı konusunda sorunlara sahiptir [11].

Literatürde mutasyon analiziyle ilgili çok fazla çalışma bulunmamaktadır. Konuyla ilgili çalışmalarda genellikle, mutant oluşturmak için operatörler geliştirilmiştir [15,10]. Fakat geliştirilen operatörlerle mutant oluşturulması, büyük modellerde zaman ve alan açısından problemlere neden olmaktadır.

Bu makalenin amacı, mutasyon analizinde zaman ve alan karmaşıklığını azaltarak, mutasyon analizini daha verimli hale getirmektir. Buradan hareketle çalışmada, modelin bütünü yerine, önemli varsayılan bölgelerini kontrol edebilecek ve bunun sonucunda verimde artış sağlayabilecek bir yöntem önerilmiştir.

Test sonucuna etki etme oranı en fazla olan bölgelerin, hata oluşmasına en uygun bölgeler olduğu varsayılmıştır. Çalışma yöntemi olarak öncelikle, Fourier dönüşümü kullanarak, modelin sonucuna en fazla etki eden bölgeler belirlenmiştir. Böylece bütün bir model yerine, tespit edilen bölgelerden yola çıkılarak mutantlar oluşturulmuştur.

Önerilen yöntemi test etmek üzere, iki adet mutant kümesi oluşturulmuştur. İlk kümede, literatürde yer alan "geçiş yönünün değiştirilmesi, geçiş eksikliği, girdinin değişimi ve çıktının değişimi" gibi mutasyon operatörleri kullanılarak, modelin tamamından mutantlar oluşturulmuştur. Diğer kümede ise literatürde yer alan "geçiş yönünün değiştirilmesi, geçiş eksikliği, girdinin değişimi ve çıktının değişimi" gibi mutasyon operatörleri kullanılarak, Fourier dönüşümü ile sonuca en fazla etki ettiği tespit edilen bölgelerden mutantlar oluşturulmuştur. Son olarak bu iki küme, W metoduyla oluşturulan test kümeleriyle analiz edilmiştir. Test kümesi üretme aracı olarak W metodun kullanılmasının nedeni, bu metodun literatürde yer alan oparatörlerle yaratılmış tüm mutantları yakalayabilme özelliğidir [8]. Mutant kümelerinin, W metodu ile test edilmesinden çıkan sonuçlar grafik ve T-Test kullanılarak değerlendirilmiştir.

Çalışmada Bölüm 2'de, Mutasyon Analizine değinilmiştir. Bölüm 3'te, Sonlu Durum Makinesi Testleri açıklanmıştır. Daha sonra Bölüm 4'te, W Yöntemi incelenmiştir. Devamında Bölüm 5'te, makalenin ana konusu olan Fourier Açılımı Yöntemiyle Mutant Yaratılması ele alınmıştır. Son olarak Bölüm 6'da, elde edilen veriler test edilerek, değerlendirilmiştir.

2 Mutasyon Analizi

Mutasyon analizi, test faaliyetlerini desteklemek için yapay hataları kullanma fikrinden yola çıkmaktadır [13]. Mutasyon, tipik olarak test takımlarının yeterliliğini değerlendirmek, test senaryolarının oluşturulmasına rehberlik etmek ve deneyleri desteklemek için kullanılmaktadır [16]. Mutasyon analizi, tipik programlama hatalarını temsil eden orijinal programda basit sözdizimsel modifikasyonlar (mutantlar) üretir [2]. Örneğin, bir mutasyon operatörü, orijinal programdaki bir aritmetik operatörünü, başka bir operatörle değiştirir; eğer bir test kümesi bir mutantı orijinal programdan ayırabilirse, mutantın öldürüldüğü söylenir. Aksi taktirde, mutanta canlı mutant denir. Bir mutant canlı kalabilir, çünkü orijinal programa eşdeğerdir veya test kümesi mutantı öldürmek için yetersizdir. Eğer bir test kümesi yetersizse, eşdeğer olmayan canlı mutantı öldürmek için test senaryoları eklenerek iyileştirilebilir. Mutasyon analiziyle bir test kümesinin yeterliliği, öldürülen mutant sayısının, eşdeğer olmayan mutant sayısına oranıyla belirlenir. Bu oran aynı zamanda mutasyon skoru olarak da adlandırılır.

Mutasyon analizi programlardaki küçük değişikliklerin karmaşık hataları ortaya çıkarmak için yeterli olduğunu varsaymaktadır. Mutasyon analizi sadece program kodunda değil aynı zamanda tasarım modellerinde, sistem özellikleri ve sistem arayüzlerinde de kullanılmıştır.

FSM'ler (Sonlu Durum Makinesi) test edilirken, sıklıkla mutasyon analiz tekniği tercih edilmektedir. Ancak literatüre baktığımızda, FSM'nin mutasyon analizi ile ilgili çok az yayın bulunduğu tespit edilmiştir. Test kümesini kapsaması bakımından, mutantların her biri aynı önem derecesine sahip değildir. Bu çalışmada, çıkış ve durum değerlerinde hata olma oranı en yüksek bölgelerin, mutantlar oluşturularak analiz edilmesi amaçlanmıştır.

2002 yılındaki bir makalede [15] ifade edildiği üzere, mutasyon operatörleri şöyledir: Geçiş eksikliği, hatalı başlangıç durumu, çıktının değişmesi, fazladan durum, çıktının eksikliği, çıktının fazlalığı, durum eksikliği, durumun değişmesi. 2009 yılındaki bir başka makaleye göreyse [10] ise mutasyon operatörleri şu şekildedir: Geçişin yönünün değişmesi, geçişin yok olması, geçişin tekrarlanması, girdinin değişmesi, girdinin eksikliği, çıktının değişmesi, çıktının yok olması, çıkış durumunun değişmesi, çıkış durumunun tekrarı, başlangıç durumunun değişmesi, başlangıç durumunun tekrarı. Bildiğimiz kadarıyla, sonraki yıllarda bu konuda kapsamlı bir çalışma bulunmamaktadır.

3 Sonlu Durum Makinesi Testi

Sonlu durum makinesi (FSM) modeli, ayrık giriş ve çıkışlara sahip matematiksel bir modeldir. Bu model donanım ve yazılım sistemlerinin modellenmesinde kullanılır. Yazılım ve donanım sistemlerinin örnekleri metin editörü, derleyiciler ve senkron ardışık devrelerdir. Sayısal bilgisayarlar da bu modele uygun sistemler olarak görülebilir. Bu nedenle, bilgisayar bilimi ve mühendisliği için sonlu durumlu makineleri önemli bir modeldir [3,7,4].

Bir sonlu durum makinesi M = $\langle$I, O, S, f, g, $\sigma\rangle$ 6-lısı' ndan oluşur. Burada I, giriş sembolleri sonlu kümesini; O, çıkış sembolleri sonlu kümesini; S, sonlu

durumlar kümesini; f: S $\times$ I $\rightarrow$ S bir sonraki durumu belirleyen geçiş fonksiyonunu; g: S $\times$ I $\rightarrow$ O, çıkış fonksiyonunu ve σ ise makinenin başlangıç durumunu göstermektedir.

Hem devreleri hem de yazılımı temsil eden biçimsel bir modelleme tekniği olarak Sonlu Durum Makinesi (FSM), testlerde yaygın olarak kullanılmaktadır. FSM testi yaygın olarak çalışılan bir konudur ve literatürde W, Wp, UIO, UIOv, DS, HSI ve H gibi birçok test kümesi üretme yöntemi bulunmaktadır.

Test ile ilgili bazı önemli tanımlar aşağıda verilmiştir.

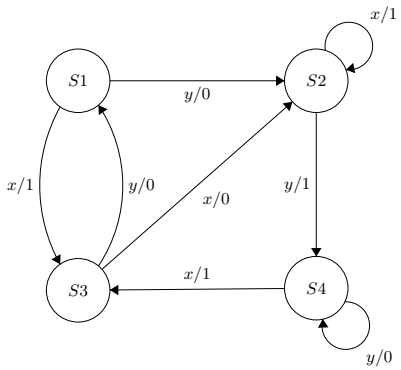
Test Durumu: Belirli bir program yolunu çalıştırmak ya da bir gereksinim ile uyumluluğunu doğrulamak gibi belirli bir amaç veya test koşulu için geliştirilen, bir dizi girdi değeri, test öncesi yürütülmesi gereken önkoşullar, test sonrası oluşması beklenen sonuçlar ve koşullar bütünü.

Test Kümesi: Bir sistem veya bileşeni test etmek için oluşturulmuş test senaryoları kümesi. Öyle ki, bir test durumu için art koşul olan bir durum bir diğeri için ön koşuldur.

Diğer yandan, yazılım çeşitliliği ve boyutundaki artış, testlerde "arıza tespit oranı, arıza ortaya çıkarma potansiyeli, test paketi boyutu, test durumu uzunlukları ve test için harcanan süre" gibi konularda, daha iyi test paketleri üretilebilmesini gerektirmektedir.

Proje kapsamında, sonlu durum makinelerinin testinde W metodun tercih edilmesinin nedeni; yukarıda ifade edilen hemen hemen tüm test üretme yöntemlerinde bu metodun değiştirilmiş biçimlerde kullanılıyor olmasıdır[11]. Aynı zamanda W metot, oluşturulabilecek tüm mutantları yakalayabilecek yapıya sahiptir. Bu sayede, standart biçimde oluşturulan mutantlarla, Fourier dönüşümü kullanılarak oluşturulan mutantları, W metodu tarafından oluşturulan test kümesini kapsaması bakımından, kıyaslama imkanı elde edilmiştir.

Çalışma kapsamında yer verilecek örneklerin FSM'si Şekil 1'de verilmiştir:



Şekil 1. örnek FSM

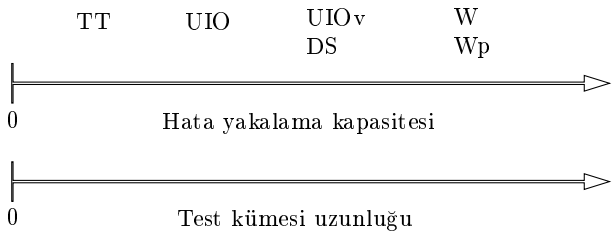
4 W Yöntemi

W yöntemi, oluşturulabilecek tüm mutantları yakalayabilecek yapıya sahiptir [8]. Bu sayede, bütün modelden oluşturulan mutantlarla, Fourier dönüşümü kullanılarak oluşturulan mutantlar, W metodu tarafından oluşturulan test kümesini kapsaması bakımından, kıyaslanmıştır.

Tablo 1'de, literatürde yaygın olarak kullanılan bazı test üretme metotlarının hata yakalama kapasiteleri bakımından karşılaştırılması verilmiştir. Şekil 2'de ise aynı metotların, hata yakalama kapasitesi ile test kümesi uzunluklarının karşılaştırılması yer almaktadır.

Tablo 1. Test Üretme Metotlarının Hata Yakalama Kapasitesi [8]

	Çıkış Hatalarını Yakalama	Geçiş Hatalarını Yakalama	Yeni Durum Hatalarını Yakalama	Eksik Durum Hatalarını Yakalama
Transition Tour	Her Zaman	-	-	-
DS-Method	Her Zaman	Her Zaman	-	Her Zaman
W-Method	Her Zaman	Her Zaman	Her Zaman	Her Zaman
UIO-Method	Her Zaman	Yaklaşık	Yaklaşık	Yaklaşık



Şekil 2. Test Üretme Metotlarının Hata Yakalama Kapasitesi ile Test Kümesi Uzunluğunun Karşılaştırması [8]

W yöntemi, iki ana bölümden oluşmaktadır. Bunlar, sonraki durumu test etmeye yarayan "W-set" bölümü ve iki durum arasındaki geçişi test etmeye yarayan "geçiş kapsama kümesi"(GKK)'dir.

Bu çalışmada, sonraki durumu test etmek üzere W-set bölümü olduğu gibi kullanılmıştır. Şekil 1'deki örnek FSM'den oluşturulan W-set, şöyledir: {yy, y, x}.

Geçiş kapsama kümesi, tüm geçişleri ziyaret etmek amacıyla yaratılır. Bunu yapmak için çizgeden ağaç elde edilir. Bu ağacın bütün uçlarına giden yollar bulunur. Bu yolların girdi değerleri kümesi, geçiş kapsama kümesini oluşturur. Bu çalışmada, geçiş kapsama kümesinin azaltılması hedeflenmiştir. Şekil 1'deki

örnek FSM'den oluşturulan geçiş kapsama kümesi, şöyledir: {x, y, xx, xy, yx, yy, yyx, yyy}. W yöntemince test kümesi, GKK ile W-set'in çarpımından oluşmaktadır:

$$W\,Metodu = GKK \times W\,set = \{x, y, xx, xy, yx, yy, yyx, yyy\} \times \{yy, y, x\} = \{xyy, xy, xx, yyy, yy, yx, xxyy, xxy, xxx, xyyy, xyy, xyx, yxyy, yxy, yxx, yyyy, yyy, yyx, yyxyy, yyxy, yyxx, yyyyy, yyyy, yyyx\}$$

5 Fourier Açılımı Yöntemiyle Mutant Yaratma

Zaman ve alan karmaşıklığını azaltarak, mutasyon analizini daha verimli hale getirmek amacıyla çalışmada, modelin bütününden mutant oluşturmak yerine, önemli varsayılan bölgelerinden mutant yaratılarak; verimde artış sağlayabilecek bir yöntem önerilmiştir. Bu amaç doğrultusunda, modelin sonucuna etki etme oranı en fazla olan bölgelerin hata oluşma ihtimali en yüksek bölgeler olduğu varsayımıyla, modeldeki geçişler önceliklendirilmiştir. Buradan hareketle öncelikle, Fourier dönüşümü kullanılarak, modelin sonuca en fazla etki eden bölgeleri tespit edilmiştir. Böylece bütün bir model yerine, tespit edilen bölgelerden mutantlar üretilmesi hedeflenmiştir.

Mutasyon analizinde, iki adet mutasyon kümesi oluşturulmuştur. İlk kümede, bütün modelden mutantlar oluşturulmuştur. Diğer kümede ise Fourier dönüşümü ile tespit edilen ve sonuca en fazla etki eden bölgelerden mutantlar oluşturulmuştur. Son olarak bu iki yöntem, W metoduyla oluşturulan test takımlarıyla analiz edilmiştir. Bu sayede sonuca en fazla etki eden bölgeler, başarımda önemli bir düşüş olmadan, daha az sayıda mutantla analiz edilebilmiştir.

5.1 Fourier Analizi

Bu bölümde, ikili işlevlerin Fourier analizinin temel kavramları tanıtılmaktadır. Her ne kadar ikili işlevlerin Fourier analizi, son on yılda matematik ve mühendislik alanında ilgi çekici bir konu olmasına rağmen, henüz çok az pratik uygulamaya sahiptir. İkili işlevler genellikle $f : \mathbb{B}^n \to \mathbb{B}$ olarak ifade edilir. Burada $\mathbb{B}$ = {Doğru, Yanlış}'dır. Fourier analizinin bir gereği olarak, *Doğru* ve *Yanlış* yerine, genellikle -1 ve 1 değerleri kullanılmaktadır ($f : \{-1, 1\}^n \to \{-1, 1\}$). Analize ilişkin temel tanım ve teoremler kanıtlar olmaksızın aşağıda verilmiştir. Diğer teoremler, kanıtlar ve örnekler için okuyucu O'Donnell [12] ve Wolf'un [5] çalışmalarına bakabilir.

Teorem 1 (Fourier Açılımı). *Her bir fonksiyon $f : \{-1, 1\}^n \to \mathbb{R}$ Fourier açılımı ile benzersiz bir şekilde ifade edilebilir.*

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S)\chi_S$$

Burada $\hat{f}(S)$ Fourier katsayılarını, $\chi_S = \prod_{i \in S} x_i$ ise Parity fonksiyonunu ifade etmektedir.

Tanım 1 (İç Çarpım). *İç çarpım aşağıdaki formülle bulunur:*

$$< f, g >= \frac{\sum_{x \in \{-1,1\}^n} f(x)g(x)}{2^n}$$

Fourier katsayılarıysa aşağıdaki gibi hesaplanır:

$$\hat{f}(S) =< f, \chi_S >$$

Aşağıda basit bir ikili işlev üzerinden Fourier açılımının nasıl hesaplandığı gösterilmektedir. İşlevimiz aşağıdaki gibi olsun:

$$f = a \vee b \wedge c$$

f'in doğruluk yöneyinin aşağıdaki gibi olduğu kolayca görülebilir (Y yanlışı, D doğruyu temsil etmektedir):

$$[\mathrm{Y\,Y\,Y\,D\,D\,D\,D\,D}]$$

İşlevin Fourier açılımı şu şekilde yazılabilir:

$$f = \hat{f}(\emptyset) + \hat{f}(1)a + \hat{f}(2)b + \hat{f}(3)ab + \hat{f}(4)c + \hat{f}(5)ac + \hat{f}(6)bc + \hat{f}(7)abc$$

Tanım 1 uyarınca, ilk Fourier katsayısı $\hat{f}(\emptyset)$ aşağıdaki şekilde hesaplanır:

$$\hat{f}(\emptyset) = \frac{1}{2^3}(1 + 1 + 1 - 1 - 1 - 1 - 1 - 1) = 0.25$$

Benzer şekilde ikinci katsayı aşağıdaki gibi bulunur:

$$\hat{f}(1) = \frac{1}{2^3}(1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 - 1 \cdot 1 - 1 \cdot -1 - 1 \cdot -1 - 1 \cdot -1 - 1 \cdot -1) = 0.75$$

Tüm katsayılar hesaplandığında Fourier açılımı aşağıdaki şekilde elde edilmiş olur:

$$f = 0.25 + 0.75a + 0.25b + 0.25ab + 0.25c + 0.25ac - 0.25bc - 0.25abc$$

Yukarıda oluşturulan f fonksiyonuna bakıldığında, katsayısı en yüksek bileşenlerin, fonksiyonu en fazla değiştirme özelliğine sahip olduğu görülmektedir. Bu açıdan, fonksiyonda yer alan katsayısı yüksek bileşenler, hata oluşma ihtimalinin yüksek olduğu bölgelere işaret etmektedir.

5.2 Önerilen Algoritma

Fourier dönüşümü ikili işlevler üzerinde tanımlı olduğundan, FSM öncelikle bu forma dönüştürülür. Bu dönüşüm, temelde durumların (state) ikili kodlanmasından ibarettir. Örneğin S1, 00 olarak, S2, 01 olarak ve girişler x ile y, 1 ve 0 olarak kodlanabilir.

Çıkış işlevi Denklem 1'deki gibidir. Çıkış işlevi, girdilerden ve durumlardan oluşmuştur.

$$O = (\bar{S_1} \wedge \bar{S_2} \wedge X) \vee (\bar{S_1} \wedge S_2 \wedge X) \vee (\bar{S_1} \wedge S_2 \wedge \bar{X}) \vee (S_1 \wedge S_2 \wedge X) \quad (1)$$

Doğruluk tablosu çıkış işlevi kullanılarak elde edilir. Girişler ve durumlar, Fourier katsayılarını elde etmek için kullanılır. Fourier açılımı Denklem 2'de verilmiştir:

$$O = 0 - 0.5S_1 + 0.5S_2 + 0 + 0.5X + 0 + 0 + 0.5S_1S_2X \quad (2)$$

Örnekte görüldüğü gibi, bazı katsayılar sıfırdır. Bu, ilgili bileşenlerin fonksiyon üzerinde etkisi olmadığını belirtmektedir. Bu bileşenlerle ilgili olan test senaryolarını silmek, test kümesi boyutunu ve test kümesi maliyetini azaltır. Bu nedenle bir eşik değer belirlenerek eşiğin altında kalan bileşenler ihmal edilir. Testlerin belirlenmesinde ve test senaryo sayısının azaltılmasında rol oynayan iki parametre belirlenmiştir. İlk parametre, Fourier açılımından kaç bileşenin alınacağı; ikinci parametre ise kaç geçişin alınacağıdır.

Parametrelerin Fourier açılımına uygulanmasıyla, bazı bileşenler ihmal edilir ve bunun sonucunda O fonksiyonu Denklem 3'te görüldüğü şekle indirgenir.

$$O = 0.5S_1 + 0.5S_2 + 0.5X + 0.5S_1S_2X \quad (3)$$

Denklem 3'te gösterildiği gibi, katsayılar belirlendikten sonra uygun geçişler seçilir. Tablo 2'de ilgili algoritmalar çalıştırıldıktan sonra seçilmiş olan geçişler gösterilmektedir.

Tablo 2. Bütün terimler tek tek geçişlere uygulandıktan sonra seçilmiş olan geçişleri göstermektedir.

S1	S2	X	S1	S2	Output
0	0	0	1	0	0
0	1	0	1	1	1
1	1	1	0	1	1
0	0	1	0	1	1
0	1	1	1	0	1
1	0	0	0	1	0

Fourier terimleriyle ilintili geçişleri seçmek için iki algoritma geliştirilmiştir. İlk algoritmada, kullanıcı tarafından verilen birinci ve ikinci parametre değerlerine göre, kaç adet terim ve kaç adet geçiş olacağının belirlenmesi sağlanmıştır.

Diğer algoritma ise, Fourier değişkenlerini bir şablon olarak alır. Geçişin uygun olup olmadığı, bu şablondan kontrol edilir. Desen bir ise ve herhangi bir değişiklik yoksa, program yanlış döner. Diğer yandan, eğer desen sıfır ise ve bir değişiklik varsa, program yine yanlış döner. Programın sonunda doğru döner.

Sadece seçilen geçişlerin mutasyona uğratılmasıyla elde edileceke mutant sayısında önemli azalma sağlanır. Böylelikle mutasyon analizinin hesaplama yükünün azalması beklenir. Bunun yanı sıra, performans olarak diğer mutasyon analiz operatörlerinden daha yüksek başarım elde edilmiştir.

6 Değerlendirme

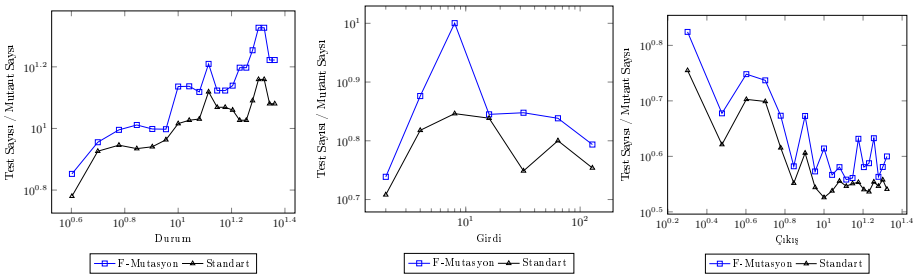
Literatürde varolan mutasyon analizi ile çalışmada önerilen mutasyon analizi, W metodun verdiği sonuçlara bakılarak karşılaştırılmış ve önerilen mutasyon analizinin başarımı gözlenmiştir.

Literatürde varolan mutasyon kümesiyle ile çalışmada önerilen mutasyon kümesinin karşılaştırılmasında, "geçiş yönünün değiştirilmesi", "geçiş eksikliği", "girdinin değişimi" ve "çıktının değişimi" hata sınıflarına göre mutantlar üretilmiştir.

Bu çalışmada, bir test kümesinin belirli bir hedefe uygun olarak azaltılması önerilmektedir. Ancak geçerlilik tehdidi açısından, test kümesindeki bu azalmanın istenen sonucu veremediği durumlar olabilir. Başka bir deyişle, "çıktıyı en çok etkileyen yol" kavramı her durumda geçerli olmayabilir. Değerlendirme sonuçları olumlu görünse de, bu tehdidi ortadan kaldıracak formel bir yaklaşım veya kanıt bu çalışmada geliştirilmemiştir.

FSM'ler, KISS Generator 0.8 [1] programı ile, durum, çıktı ve giriş değerleri ayrı ayrı değiştirilerek rastgele üretilmiştir. Her bir FSM 5'er defa üretilmiş ve çıkan değerlerin ortalaması alınarak test sonucu elde edilmiştir. Bu işlem 300 defa tekrarlanmıştır. Yani, toplamda 1500 adet FSM yaratılmıştır. Literatüre bakıldığında, çalışmada yaratılan FSM miktarının, geçerlilik tehdidi oluşturmadığı görülmektedir. Benzer şekilde, değerlerin rastgele oluşturulması da bir diğer geçerlilik tehdidini ortadan kaldırmaktadır.

FSM'ler, W metodunun gerektirdiği şekilde *indirgenmiş*, *deterministik*, *tam bağıntılı* ve *tam belirtilmiş*'tir. Burada, karşılaştırma kriterlerinin tamamının kullanıldığı üç adet grafiğe yer verilmiştir.



Şekil 3. Durumun, Girdinin, Çıkışın Değişimine Göre Test Sayısı / Mutant Saysı

Şekil 3'ün sol paneline bakıldığında, standart ve önerilen yöntemlerin mutant başına kapsanan test sayısı duruma bağlı olarak paralel değiştiği ancak önerilen yöntemin başarımının daha yüksek olduğu görülmektedir.Burada bakılan ana nokta, her bir mutantın kapsadığı test sayısıdır. Yani önerilen yöntem,

daha fazla testi kapsamaktadır. Bu da önemli oranda verim artışı sağlamıştır. Şekil 3'ün ikinci kısmına bakıldığında, değerlerin bazı noktalarda paralel olmadığı görülmektedir. Genel olarak, önerilen metodun grafiksel başarımı daha yüksektir. Yani önerilen metot, standart metoda göre daha fazla testi kapsamaktadır. Grafiğin ilk yarısında, önerilen metodun pik yaptığı dikkat çekmektedir. Grafiğin yarısında ise önerilen metot ile standart metot hemen hemen aynı noktada yer almaktadır. Ancak grafiğin devamında ve genelinde, önerilen metodun, standart metoda göre daha fazla testi kapsadığını söylemek mümkündür. Şeklin son kısmındaysa, değerlerin genellikle paralel olduğu görülmektedir. Çıkış değişimi bakımından, önerilen metodun grafiksel başarımı daha yüksektir. Dolayısıyla grafiğin genelinde, önerilen metodun başarımı, standart metoda göre daha yüksek sonuçlar vermiştir.

7 Sonuç

Çalışmada, devre ve yazılım testlerinde yaygın kullanıma sahip bir analiz tekniği olan mutasyon analizi için mutant kümesinin küçültülmesine yönelik bir yöntem geliştirilmiştir. Bunun nedeni, FSM (Sonlu Durum Makinesi) test edilirken kullanılan mutasyon analiz yöntemlerinin, çok fazla mutant oluşturmasıdır. Bu problemden yola çıkarak projede, çıkış ve durum değerlerinde hata olma oranı en yüksek bölgelerin, mutantlar oluşturularak analiz edilmesi amaçlanmıştır. Çalışmada, test sonucuna etki etme oranı en fazla olan bölgelerin, hata oluşmasına en olası bölgeler olduğunu varsayılmıştır.

Buradan hareketle öncelikle, Fourier dönüşümü kullanarak modelin sonuca en fazla etki eden bölgeleri tespit edilmiştir. Böylece bütün bir model yerine, tespit edilen bölgelerden yola çıkılarak mutantlar oluşturulmuştur. Bu sayede, her bir geçiş için yapılması gereken değişiklikler, yalnızca Fourier dönüşümünün seçtiği geçişler üzerinden yapılmış ve kapsama oranı artırılmıştır. Bunun sonucunda mutant sayısı azalmıştır.

Önerilen mutasyon analizini test etmek üzere, iki adet mutasyon kümesi yaratılmıştır. İlk kümede, standart mutant analiz prosedürü takip edilmiştir. Diğer kümede ise Fourier dönüşümü ile tespit edilen ve sonuca en fazla etki eden bölgelerden elde edilen mutantlarla analiz yapılmıştır. Son olarak bu iki yöntem, W metot ile oluşturulan test takımlarıyla, mutasyon analizlerinin kapsama alanları bakımından karşılaştırılmıştır. Test kümesi üretme aracı olarak W metodun kullanılmasının nedeni, bu metodun varolan tüm mutantları yakalayabilme özelliğidir.

Literatürde varolan standart mutasyon analiz yöntemleri ile çalışmada önerilen mutasyon analiz yöntemi, geçiş yönünün değiştirilmesi, geçiş eksikliği, girdinin değişimi ve çıktının değişimi sınıflarına göre yapılmıştır. Karşılaştırmada önemsenen nokta, her bir mutantın kapsadığı test sayısıdır. Çalışmada, karşılaştırma kriterlerinin tamamının kullanıldığı üç adet grafiğe yer verilmiştir.

Sonuca en fazla etki eden bölgeler, daha az sayıda mutantla ve daha yüksek başarımla analiz edilebilmiştir. Bunun sonucunda, mutan sayısı azalmasına rağmen, mutantın test kapsama oranı açısından daha iyi bir sonuç elde edilmiş-

tir. Sonuç olarak, önerilen mutasyon analiz yöntemi, önemli oranda verim artışı sağlayarak, standart metotlara oranla daha iyi başarım göstermiştir.

Kaynakça

1. KISS Generator 0.8, https://ddd.fit.cvut.cz/prj/Circ_Gen/index.php?page=kiss
2. Aichernig, B.K., B, M.T.: Learning from Faults : Mutation Testing in Active Automata Learning (2017). https://doi.org/10.1007/978-3-319-57288-8
3. Belli, F., Beyazıt, M., Endo, A.T., Mathur, A., Simao, A.: Fault domain-based testing in imperfect situations: a heuristic approach and case studies. Software Quality Journal **23**(3), 423–452 (2015)
4. Damasceno, Carlos Diego Nascimento ve Masiero, P.C.v.S.A.: Evaluating test characteristics and effectiveness of fsm-based testing methods on rbac systems. Proceedings of the 30th Brazilian Symposium on Software Engineering - SBES '16 (2016)
5. De Wolf, R.: A brief introduction to fourier analysis on the boolean cube. Theory of Computing, Graduate Surveys **1**(1-20), 15 (2008)
6. Endo, Andre Takeshi ve Simao, A.: Evaluating test suite characteristics, cost, and effectiveness of fsm-based testing methods. Information and Software Technology (2013). https://doi.org/10.1016/j.infsof.2013.01.001
7. Fragal, V.H., Simao, A., Mousavi, M.R., Turker, U.C.: Extending HSI Test Generation Method for Software Product Lines. The Computer Journal (2018)
8. König, H.: Protocol Engineering. Computer Science, Springer Berlin Heidelberg (2012), https://books.google.com.tr/books?id=Sr3NdA1PQc4C
9. Lee, David ve Yannakakis, M.: Principles ve methods of testing finite state machines (1996)
10. Li, J.H., Dai, G.X., Li, H.H.: Mutation analysis for testing finite state machines. 2nd International Symposium on Electronic Commerce and Security, ISECS 2009 **1**, 620–624 (2009). https://doi.org/10.1109/ISECS.2009.158
11. Mathur, A.P.: Foundations of software testing (2005)
12. O'Donnell, R.: Analysis of boolean functions. Cambridge University Press (2014)
13. Papadakis, M., Just, R.: Special issue on Mutation Testing. Information and Software Technology **81**, 1–2 (2017). https://doi.org/10.1016/j.infsof.2016.08.003, http://dx.doi.org/10.1016/j.infsof.2016.08.003
14. Papadakis, M., Kintis, M., Zhang, J., Jia, Y., Traon, Y.L., Harman, M.: Mutation Testing Advances: An Analysis and Survey. Advances in Computers **112**, 275–378 (2019). https://doi.org/10.1016/bs.adcom.2018.03.015
15. Pinto Ferraz Fabbri, S., Delamaro, M., Maldonado, J., Masiero, P.: Mutation analysis testing for finite state machines pp. 220–229 (2002). https://doi.org/10.1109/issre.1994.341378
16. Zhang, J., Zhang, L., Harman, M., Hao, D., Jia, Y., Zhang, L.: Predictive Mutation Testing. IEEE Transactions on Software Engineering **14**(8) (2018). https://doi.org/10.1109/TSE.2018.2809496

Anti-Social: Bireylerin Sosyal Medya Bağımlılığını Azaltmak Amacıyla Düşünülmüş Bir Ciddi Oyun

Ulaş Güleç[1], Murat Yılmaz[2], Can Koral Adalı[2] ve Veysi İşler[1]

[1] Hasan Kalyoncu Üniversitesi, Bilgisayar Mühendisliği Bölümü, Gaziantep, Türkiye
ulas.gulec,veysi.isler@hku.edu.tr
[2] Çankaya Üniversitesi, Bilgisayar Mühendisliği Bölümü, Ankara, Türkiye
myilmaz@cankaya.edu.tr, cankorala@hotmail.com

Özet. Günümüzde, teknolojinin hızlı bir şekilde gelişmesinden dolayı dijital marketlerde kişilerin kullanımına sunulan çok sayıda mobil uygulama bulunmaktadır. Farklı amaçlar doğrultusunda geliştirilen bu uygulamaların indirilme oranına ve kullanım sıklığına bakıldığında; Facebook, Twitter ve Instagram gibi sosyal medya uygulamalarının, diğer uygulamalara göre daha çok indirildiği ve kullanıldığı ortaya çıkmaktadır. Kullanıcıların bu uygulamaları hangi sıklıkta kullandığı detaylı bir şekilde incelendiğinde ise, sosyal medya uygulamalarının birçok kişide bağımlılık yarattığı ve bu bağımlılığın kişileri hem fiziksel hem de psikolojik açıdan olumsuz bir şekilde etkilediği gözlemlenmiştir. Bu tip olumsuzlukları engelleyerek kişilerin yaşam kalitesini düşüren bu etkenleri ortadan kaldırmak, toplumlarda hem fiziksel hem de zihinsel bakımdan daha sağlıklı bireylerin yaşamasında ve yetişmesinde önemli bir rol oynamaktadır. Bu kapsam doğrultusunda bu çalışma, kişilerin sosyal medya bağımlılığını düşürmeyi amaçlayan bir ciddi oyun tasarlamayı hedeflemektedir. Android işletim sistemi içeren telefonlar ve tabletler için tasarlanan bu oyun, kişilerin sosyal medya kullanımlarını takip ederek sosyal medya uygulamalarını kullanmadıkları her an için kişilere farklı tiplerde ödüller vermektedir. Geliştirilen bu oyunun kişiler üzerindeki etkisini anlayabilmek için oyun, yaşları 21 ile 26 arasında olan 5 kişi ile 4 hafta boyunca test edilmiştir. Testlerden elde edilen sonuçlara göre, tasarlanan ciddi oyunun sunmuş olduğu ödül mekanizması sayesinde kişilerin ilgisi çekilerek sosyal medya kullanımının azaldığı gözlemlenmiştir.

Anahtar Kelimeler: Ciddi Oyunlar, Sosyal Medya Bağımlılığı, Yaşam Kalitesi, Kişisel Sağlık

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Anti-Social: A Serious Game for Reducing the Social Media Addiction of Individuals

Ulas Gulec[1], Murat Yilmaz[2], Can Koral Adali[2] and Veysi Isler[1]

[1] Hasan Kalyoncu University, Department of Computer Engineering, Gaziantep, Turkey
ulas.gulec,veysi.isler@hku.edu.tr
[2] Çankaya University, Department of Computer Engineering, Ankara, Turkey
myilmaz@cankaya.edu.tr, cankorala@hotmail.com

Abstract. Nowadays, there are a large number of applications available in digital markets to individuals due to the rapid development of technology. When the download rate and usage frequency of these applications developed for different purposes are analyzed, it is observed that social media applications such as Facebook, Twitter and Instagram are more downloaded and used than other applications. When a detailed research is completed on how often users use these practices, it is seen that social media applications create dependence on many people and this dependence negatively affects individuals both physically and psychologically. Hence, eliminating the factors that reduce the quality of life of individuals by preventing these negative effects of social media plays a critical role to provide in the life and growth of healthy individuals both physically and mentally. In line with this scope, this study aims to design a serious game that aims to reduce the social media addiction of individuals. This game designed for phones and tablets with Android operating system provides different types of rewards for people at any time when they don't use social media applications. In order to figure out the impact of the game on individuals, it was tested with 5 participants between 21 and 26 years old during 4 weeks. According to the results obtained from the tests, it has been seen that the serious game designed has reduced the use of social media by attracting the attention of people due to the award mechanism.

Keywords: Serious Games, Social Media Addiction, Life Quality, Personal Health

1 Giriş

Günümüz dünyasında sosyal medya kullanımı; çocukların, gençlerin ve yetişkinlerin yapmış olduğu en yaygın aktivitelerden biri haline gelmiştir [20]. Özellikle Facebook, Twitter, Instagram ve YouTube en çok kullanılan sosyal medya platformları olarak ön plana çıkmaktadır [18,23]. Bu uygulamaların kişiler tarafından bu kadar sık bir şekilde kullanılmasındaki en büyük etken; internet teknolojisinin

son yıllarda hızlı bir şekilde gelişmesi ve kişilerin internete daha rahat, hızlı ve ucuz erişilebilmesidir [24]. Bu düşünceyi desteklemek amacıyla, Smith ve Anderson [26] tarafından yapılan bir çalışmada, Amerika'da yaşayan insanların 2018 yılındaki sosyal medya kullanımları incelenmiştir. Bu inceleme sonucunda elde edilen sonuçlara göre, internetin de gelişimi ile birlikte Amerikalıların çoğunun Facebook ve Youtube platformlarını çok sık bir şekilde kullandıkları, gençlerin ise Snapchat ve Instagram uygulamalarını çok aktif bir şekilde kullandıkları ortaya çıkmıştır. Bu kapsam doğrultusunda Ocak 2016'da yapılan diğer bir çalışmada ise kişilerin sosyal medya kullanımları incelenmiştir [5]. Bu incelemeler sonucunda elde edilen sonuçlara göre, dünyada yaklaşık olarak 7400 milyar insanın yaşadığı, bu insanlardan yaklaşık olarak 3400'ünün interneti kullandığı ve bu 3400 milyar kişinin yaklaşık olarak 2300'ünün sosyal medyayı aktif bir şekilde kullandığı bilgisi elde edilmiştir.

Genellikle kişiler, sosyal medya uygulamalarını gerçek hayattan bir nebze de olsa kurtularak deşarj olmak amacıyla kullanmayı hedeflese de aslında bu tip uygulamalar sahip oldukları içeriklerden dolayı kişilerde bağımlılık yaratarak kişileri belli bir süre sonra depresyona sürüklemektedir [22]. Başlangıçta bireyler kendilerini sosyal medyada mutlu hissederken belli bir süre sonra hem başkaları tarafından yapılan negatif içerikli paylaşımlardan hem de başkalarının yapıp da kendilerinin yapamadığı içeriklerden dolayı kendilerini hem mutsuz hem de yetersiz hissetmektedirler [27]. Üstelik bireyler birden fazla sosyal medya platformunu aynı anda kullanıyorsa bu tip istenmeyen durumların kişiler üzerindeki olumsuz etkileri daha da artmaktadır [4]. Bireylerde oluşabilecek bu tip olumsuz durumları engelleyebilmek adına bu çalışma, kişilerin sosyal medya bağımlılığını azaltmak amacıyla Android işletim sistemlerine sahip mobil cihazlarda çalışabilecek bir oyun tasarlamayı hedeflemektedir. Bu oyun, kişilerin sosyal medya kullanımlarını platform bazında takip ederek bu uygulamaları kullanmadıkları her süre için kişilere oyun içerisinde kullanabilecekleri çeşitli ödüller sunmaktadır. Bu şekilde, kullanıcıların sosyal medya kullanımlarının azalması beklenmektedir.

Bildirinin geri kalan kısmı şu şekilde oluşturulmuştur: Bir sonraki bölümde, çalışmanın alt yapısını oluşturmak için literatürdeki benzer çalışmalar detaylı bir şekilde anlatılacaktır. Üçüncü bölümde, geliştirilen mobil uygulama görseller ile birlikte detaylı bir şekilde açıklanacaktır. Bir sonraki bölümde, sistemi test etmek amacıyla çalışmaya katılan 5 kişinin oyunu oynamadan öncesi ve sonrası ile ilgili sosyal medya uygulamalarını ne kadar kullandığına dair sayısal veriler paylaşılacaktır. Son bölümde ise, çalışmadan elde edilen sonuçlar tartışılarak bildiri sonlandırılacaktır.

2 Literatür Araştırması

Oyunlar, eğlence özelliğinin yanı sıra birçok farklı alanda kişilerin gelişiminde kullanılan faydalı bir araç haline gelmiştir [15]. Özellikle eğitim alanında çok yaygın bir şekilde kullanılan oyunlar sayesinde; ilkokul [3], ortaokul [6], lise [16] ve üniversite [2] gibi farklı seviyelerde eğitim alan öğrenciler geleneksel eğitim yöntemlerine destek olacak şekilde eğlenceli yoldan bilgi seviyelerini arttırma

şansına sahiptirler. Eğitim alanına ek olarak oyunlar; askeri [10], spor [14] ve sağlık [19] alanlarında da kişilerin hem bilgi hem de tecrübe seviyelerini arttırmada kullanılabilen etkin bir araç olarak ortaya çıkmaktadır.

Literatürde sağlık alanı için hem o alanda görev yapacak personelin bilgi ve tecrübe seviyesini arttıran hem de tedavi gören hastaların iyileşme sürecine destek olan birçok farklı oyun bulunmaktadır. Bu kapsamda, Kato ve diğerleri [17] tarafından tamamlanan bir çalışmada akut lösemi, lenfoma ve yumuşak doku sarkomu hastalık tiplerinden tümöre sahip olan hastaların, kanser ile ilgili bilinç seviyelerini arttırmak ve hastalık takibini yapabilmek adına bir video oyunu geliştirilmiştir. Geliştirilen bu oyun ile hastaların stres yönetimi, hastalık belirtilerine karşı ani müdahele yöntemleri ve ilaç kullanımı gibi birçok farklı konuda hastalık karşısında bilinçlenmeleri beklenmektedir. Amerika, Kanada ve Avustralya'da bulunan 34 tedavi merkezinde yaşları 13 ile 29 arasında değişen toplam 375 hastanın katılımı ile gerçekleşen bu çalışmadan elde edilen sonuçlar, geliştirilen video oyununun hastaların hastalık karşısında bilinçlenerek özellikle stres yönetiminden dolayı hastalığı yenme konusunda hastalara yardımcı olabilecek bir araç olduğunu göstermiştir. Bu alanda yapılan diğer bir çalışmada ise lösemi hastalığına sahip çocukların sosyal hayattan uzaklaşmamaları için akranlarının sosyal hayatta yaptıkları aktiviteleri geliştirilen oyun ile yapmaları amaçlanmıştır [8]. 8 yaşındaki lösemi hastalığına sahip 3 hastanın katılımı ile gerçekleşen çalışmada çocuklar geliştirilen oyun içerisinde matematik, sosyal davranışlar gibi farklı alanlarda kendilerine gelen görevleri tamamlamaya çalışmaktadır. Hem çocuklar ile hem de çocukların aileleri ile yapılan değerlendirmeler sonucunda, geliştirilen bu oyunun lösemi hastalığına sahip çocukların gelişiminde kullanılabilecek faydalı bir araç olduğu belirtilmiştir. Chiang ve diğerleri [7] tarafından yapılan diğer bir çalışmada ise tekerlekli sandalye kullanan yaşlıların el-göz koordinasyonunu ve hareket kabiliyetlerini artırmak için bir oyun geliştirilmiştir. Xbox 360 Kinect kullanılarak oynatılan oyun, biri 22 kişiden oluşan deney grubu diğeri ise 31 kişiden oluşan kontrol grubu olmak üzere toplam 53 kişi ile test edilmiştir. Testlerden elde edilen sonuçlara göre, Xbox 360 Kinect ile oynanan oyunların tekerlekli sandalye kullanan yaşlıların görsel yeteneklerini artırdığı tespit edilmiştir. Zielhorst ve diğerleri [28] tarafından tamamlanan diğer bir çalışmada ise tükenmişlik sendromuna çözüm bulmak için bir dijital oyun geliştirilmiştir. Toplamda 101 kişinin katılımı ile gerçekleşen bu çalışmada, katılımcılar oyun ve terapi, sadece terapi, sadece oyun ve son olarak hiçbir terapi olmayacak şekilde 4 farklı koşulda tedavi görmüşlerdir. Katılımcılar üzerindeki gelişmeyi anlayabilmek için terapilerden önce ve sonra olmak üzere katılımcılara 2 farklı test uygulanmıştır. Bu testlerden elde edilen sonuçlar, dijital oyunların tükenmişlik sendromu tedavilerinde geleneksel yöntemleri destekleyebilecek faydalı bir araç olduğunu göstermiştir.

Son yıllarda sosyal medya uygulamaları, kişilerin farklı amaçlar doğrultusunda zamanlarının büyük bir kısmını harcadıkları platformlar haline gelmiştir [13]. Yaş, din, dil, ırk, sosyo-ekonomik yapı gibi özellikler gözetmeksizin her türden insanın kullandığı sosyal medya uygulamalarının kullanım oranı son 10 yılda ciddi bir şekilde artmıştır [21]. Başlangıçta kişiler, sosyal medya uygula-

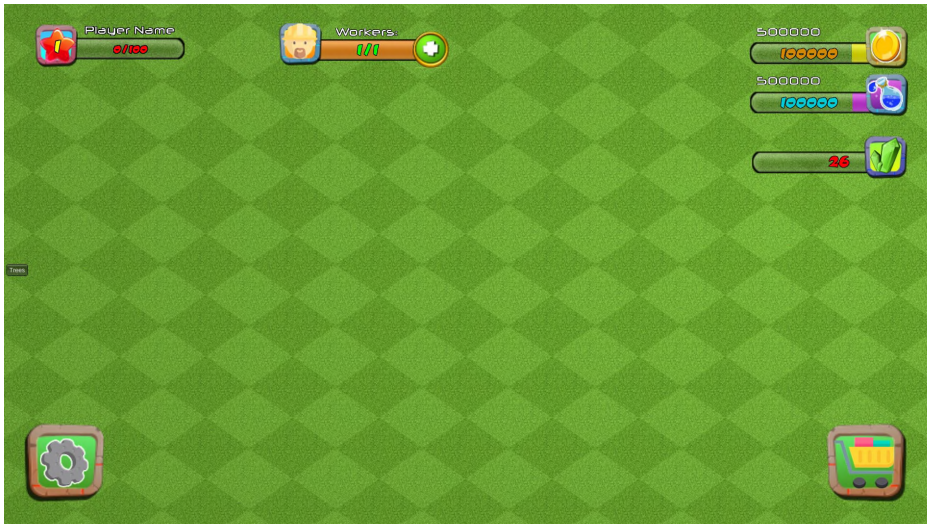
malarını deşarj olmak amacıyla kullansa da zaman içerisinde artan sosyal medya kullanımından dolayı birçok kişide hem fiziksel [25] hem de psikolojik [12] açıdan farklı problemler ortaya çıkmıştır. Üstelik bu problemler, sosyal medya kullanıcılarının çoğunun genç nesil olmasından dolayı gençlerde derin yaralar bırakarak birçok gencin geleceğini olumsuz yönde etkilemektedir [9]. Bu sebeplerden ötürü, sosyal medya bağımlılığı çağımızın yeni nesil hastalığı olarak tanımlanmaktadır [11]. Buna rağmen, yazarların bilgisine göre literatürde oyun tabanlı eğitim ve tedavi yöntemini kullanarak kişilerin sosyal medya bağımlılığını azaltmayı hedefleyen hiç bir çalışma bulunmamaktadır. Bu nedenle bu çalışma, literatürdeki bu eksikliği gidererek kişilerin sosyal medya kullanımını azaltmayı amaçlayan bir ciddi oyun tasarlamayı hedeflemektedir. Bir sonraki bölümde bu amaç doğrultusunda tasarlanan ciddi oyunu detaylı bir şekilde açıklanacaktır.

3 Yöntem

Bireyleri sosyal medya bağımlılığından eğlenceli bir şekilde kurtarmayı amaçlayan *Anti-Social*, çok oyuncu ile oynanabilen bir mobil oyundur. Bu oyunda oyuncular, sosyal medya uygulamalarını daha az bir şekilde kullanarak oyun içerisinde tanımlanmış para birimlerinden farklı miktarlarda paralar kazanabilmektedirler. Daha sonrasında oyuncular, kazandıkları paralar ile kendi köylerini/çiftliklerini kurarak haftada bir kez düzenlenen en iyi köy/çiftlik yarışmasına katılır ve haftanın en iyi tasarlanmış köy/çiftlik ödülünü alma şansına sahip olurlar. Sosyal medya uygulamalarını diğer oyunculara göre daha az kullanan oyuncu, oyun içerisinde daha fazla para kazanacağı için daha güzel görünüme sahip oyun içi ürünleri kullanarak daha güzel bir köy/çiftlik kurabilir. Böylece, sosyal medya uygulamalarını daha az kullanan oyuncular, diğer oyunculara göre oyun içerisinde daha avantajlı bir konuma gelmektedir. Bu şekilde, kişileri farklı bir alanda motive ederek kişilerin sosyal medya bağımlılığını azaltmak hedeflenmektedir.

Öncelikle oyuncular, oyunu oynayabilmek için sisteme kayıt olmak zorundadır. Kayıt işleminden sonra sistem, kayıt yaptıran her oyuncu için oyuncuya özel bir hesap oluşturmaktadır. Hesabı sistemde tanımlanan oyuncu oyuna giriş yaptığı zaman ilk olarak oyunun Şekil 1'de gösterilen ana sayfası ekrana gelir.

Şekil 1'de görüldüğü üzere oyunun ana sayfası kareler ile birbirinden ayrılmış örgü şeklinde bir zemine sahiptir. Zemin üzerinde bulunan bu karelere oyuncular oyun içindeki ürünleri yerleştirerek kendi köylerini/çiftliklerini tasarlar. Bu ürünlerin oyuncular tarafından satın alınabilmesi için öncelikle oyuncuların sosyal medya uygulamalarını kullanmayarak para kazanması gerekmektedir. Ekranın sağ üst köşesinde de görüldüğü gibi oyunda altın, güç ve kristal olmak üzere 3 farklı para tipi bulunmaktadır. Oyuncular, Facebook'a girmedikleri her süre için altın, Twitter'a girmedikleri her süre için güç ve Instagram'a girmedikleri her süre için ise kristal kazanmaktadırlar. Kazanılan para miktarı uygulamanın kullanılmadığı saat miktarı ile doğru orantılı bir şekilde artmaktadır. Oyuncular, sosyal medya uygulamalarına girmedikleri her saat için ilgili para biriminden 100 birim para kazanırlar.



Şekil 1. Anti-Social Ana Sayfası



Şekil 2. İşçi Evi

Anti-Social seviye bazlı bir oyun olup, oyuncuların daha güzel görünüme sahip oyun içi ürünlerini kendi köylerine/çiftliklerine alabilmesi için oyun içerisinde ilerleyerek yeni seviyelere ulaşması gerekmektedir. Ekranın sol üst köşesinde yer alan bardan oyun içindeki seviyelerini takip edebilen oyuncular, yeni seviyeye ulaştıkları zaman o seviyeye ait oyun içi ürünlerini yeterli miktarda paraları varsa alarak kendi köylerine/çiftliklerine ekleyebilirler. Oyuncuların oyun içerisindeki seviyelerini arttırabilmesi için kendi köylerine/çiftliklerine oyun içi ürünlerinden kurması gerekmektedir. Her ürünün seviye arttırma puanı farklı olup oyuncular ilgili ürünü kendi köylerine/çiftliklerine kurduklarında o ürünün seviye arttırma puanı oyuncunun seviye puanına eklenmektedir. Buna ek olarak, oyuncuların oyun içerisindeki ürünleri kendi köylerine/çiftliklerine kurabilmesi için yeterli miktarda insan gücüne sahip olması gerekmektedir. İlerleme barının yanında bulunan bardan takip edilebilen insan gücü için oyun, başlangıçta her oyuncuya bir işçi vermektedir. Sahip olunan işçi sayısı, aynı anda köye/çiftliğe kurulabilecek olan oyun içi ürün sayısını da göstermektedir. Çok işçiye sahip olan oyuncu aynı anda birden fazla oyun içi ürününü kendi köyüne/çiftliğine ekleyerek daha hızlı bir şekilde beklenen köy/çiftlik tasarımına sahip olabilir. Oyuncuların köyündeki/çiftliğindeki işçi sayısını arttırmak için Şekil 2'de gösterilen *İşçi Evi*'ni kendi köyüne eklemesi gerekmektedir. Köye/Çiftliğe eklenen her ev için köyün/çiftliğin sahip olduğu insan gücü 1 artmaktadır. Bu özelliğe ek olarak, işçi evleri belirli sürelerde kristal üreterek oyuncuların kristal kazanmalarına da yardımcı olmaktadır.

Oyuncular, oyun içi ürünlerini satın alarak bu ürünleri kendi köylerine/çiftliklerine kurabilmek için ana sayfada sağ alt köşede bulunan alışveriş sepetine basarlar. Sepete basıldıktan sonra Şekil 3'te gösterilen alışveriş sayfası ekrana gelir.

Alışveriş sayfasında da görüldüğü üzere alışveriş sepetinde 5 farklı ürün kategorisi bulunmaktadır:

1. **Store:** Store kategorisinde oyuncular, sahip oldukları paraları kullanarak çeşitli farklı sayılarda insan gücüne sahip olabilirler. İnsan gücündeki bu sayı oyuncunun sahip olduğu seviyeye göre değişmektedir.
2. **Production:** Bu kategoride oyuncular, sahip oldukları paralar ile para üretimine katkı sağlayacak tesisler kurabilirler. Oyun içerisinde altın kazanımını arttırmak amacıyla tasarlanan altın madeni, güç kazanımını arttırmak amacıyla tasarlanan güç ünitesi ve son olarak kristal kazanımını arttırmak amacıyla tasarlanan kristal suları olmak üzere toplam 3 tesis bulunmaktadır. Bu tesisler belirli sürelerde söz konusu kaynakların üretimini yaparak oyuncunun köy/çiftlik kurmasına yardımcı olurlar. Kurulan tesisin etki miktarı ise oyuncunun seviyesi yükseldikçe sahip olacağı bina ek onarım hakkı ile arttırılabilmektedir.
3. **Wall:** Bu kategori oyuncuya çevresel tasarım yapabilmesi için farklı tasarımlara sahip duvar ve çit modelleri sunmaktadır. Böylelikle oyuncular, tasarladıkları köyün/çiftliğin içerisinde kurmuş olduğu alanların etrafını çevreleyerek daha düzgün bir görüntü elde edebilirler.
4. **Misc:** Oyunun görsel açıdan sunmuş olduğu en önemli kategorilerden biri olan Misc kategorisi içerisinde farklı tasarımlara sahip bina ve kule modelleri



Şekil 3. Alışveriş Menüsü

içermektedir. 5. seviyeden sonra oyuncuların kullanımına açılan bu kategori sayesinde oyuncular, köy/çiftlik içerisinde güzel görünüme sahip binaları ve kuleleri kendi alanlarına ekleyebilirler.

5. **Ambient:** Bu kategoride oyuncular, köylerindeki/çiftliklerindeki yeşil alan sayısını arttırmak amacıyla farklı türlerdeki ağaç ve bitkileri kendi bölgelerine eklerler. Toplamda 8 farklı türde ağaç ve bitki içeren bu kategoride oyuncunun oyun içerisindeki seviyesine göre çiçeklerin ve ağaçların farklı renk tonları da seçilebilir hale gelmektedir.

Yukarıdaki maddelerde de görüldüğü üzere oyun, bireylerin ilgisini çekebilmek adına birçok farklı oyun içi ürüne sahiptir. Kullanıcılar sosyal medya uygulamalarını az kullanarak bu oyunda ilerleyip Şekil 4'te de gösterildiği gibi birçok farklı renkte ve türde oyun içi ürünü kendi köylerinde/çiftliklerinde kullanma şansına sahiptirler.

4 Deneyler ve Analizler

Geliştirilen oyunun kişilerin sosyal medya bağımlılığını azaltıp azaltmadığını anlayabilmek için oyun, yaşları 21 ile 26 arasında değişen 5 katılımcı ile test edilmiştir. İlk olarak katılıcımların telefonlarına, katılımcıların sosyal medya uygulamalarını hangi sıklıkta kullandıklarını takip edebilmek için bir uygulama kurulmuştur. Yüklenen bu uygulama, 24 saatlik dilimlerde kullanıcıların Facebook, Twitter ve Instagram uygulamalarını ne kadar kullandıkları bilgisini ayrı ayrı olacak şekilde merkezi bir sunucuya göndermektedir. Dakika bazında sunucuda



Şekil 4. Başlangıç Köy/Çiftlik Görünümü

tutulan bu bilgi toplama işlemi, 2 hafta boyunca devam etmiştir. Birinci gün pazartesi ve son gün pazar olacak şekilde ayarlanan süre zarfında kullanıcıların Facebook, Twitter ve Instagram uygulamalarını ortalama kullanım süreleri Tablo 1'de verilmiştir.

Tablo 1. Katılımcıların Sosyal Medya Uygulamalarını Ortalama Kullanım Süreleri

	Facebook	Twitter	Instagram
1. Katılımcı	72.3	154.8	178.4
2. Katılımcı	68.9	202.5	147.3
3. Katılımcı	28.7	239.2	102.1
4. Katılımcı	48.5	165.4	212.8
5. Katılımcı	98.6	103.7	88.4
Ortalama	**63.4**	**173.1**	**145.8**

2 haftalık veri toplama işleminden sonra, katılımcıların telefonlarına *Anti-Social* oyunu yüklenerek katılımcılardan 1 ay boyunca bu oyunu oynamaları beklenmiştir. *Anti-Social* oyunu katılımcıların telefonlarına yüklendikten sonra

da 1 aylık süre boyunca 24 saatlik dilimlerde kullanıcıların Facebook, Twitter ve Instagram uygulamalarını ne kadar kullandıkları bilgileri takip edilmeye devam edilmiştir. 1 ay boyunca toplanan bu bilgiler, haftalık bazda ortalama kullanım süresi olarak Tablo 2'de verilmiştir.

Tablo 2. 1 Ay Boyunca Katılımcıların Sosyal Medya Uygulamalarını Ortalama Kullanım Süreleri

	I			**II**			**III**			**IV**		
	F	**T**	**I**	**F**	**T**	**I**	**F**	**T**	**I**	**F**	**T**	**I**
1. Katılımcı	38.7	93.4	90.9	40.1	87.8	82.3	52.3	92.1	101.3	45.4	98.6	100.5
2. Katılımcı	22.3	151.7	108.2	29.4	132.4	86.1	25.8	144.5	97.3	30.2	167.3	103.9
3. Katılımcı	12.8	117.1	100.8	11.4	105.3	97.7	15.3	162.5	99.8	11.6	159.7	115.6
4. Katılımcı	32.5	77.6	198.5	34.2	82.3	171.2	38.9	74.2	186.6	41.4	51.6	191.1
5. Katılımcı	27.1	52.8	45.5	25.3	47.6	42.3	19.6	44.7	48.9	17.4	39.2	37.6
Ortalama	**26.7**	**98.5**	**108.8**	**28.1**	**91.1**	**95.9**	**30.4**	**103.6**	**106.8**	**29.2**	**103.3**	**109.7**

Tablo 2'de belirtilen veriler incelendiğinde, katılımcıların *Anti-Social* oyununu oynadıkları zaman sosyal medya uygulamalarını daha az kullandığı ortaya çıkmıştır. Özellikle birinci ve ikinci haftalarda oyun, katılımcıların daha çok ilgisini çekerek her 3 platformda da sosyal medya kullanımını azaltmıştır. Elde edilen bu sonuçları istatiksel açıdan da ispatlamak amacıyla, katılımcılardan 4 hafta boyunca toplanan veriler *bağımlı örneklem t-testi* ile test edilmiştir. İlk 2 haftalık süre boyunca elde edilen veriler ile 1 ay boyunca haftalık bazda elde edilen veriler karşılaştırıldığında, katılımcıların *Anti-Social* oyununu oynamaya başladıktan sonra sosyal medya kullanımlarında istatiksel açıdan önemli bir azalma olduğu görülmüştür.

5 Sonuçlar ve Gelecek Çalışmalar

Dijital oyunlar birçok farklı alanda kişilerin gelişiminde kullanılan etkili araçlardır [1]. Bu tipteki oyunlar, içerilerinde bulundurdukları özellikler sayesinde oyun olmayan alanlarda kullanıldıklarında kişilerin ilgisini çekerek öğretilmek istenen kavramları kişilere eğlenceli bir şekilde öğretmektedir. Bu sayede bu oyunlar, eğitim, sağlık vb. birçok farklı çalışma alanında kişilerin hem bilgi seviyelerini artırmada hem tecrübe seviyelerini artırmada hem de farkındalık kazandırmada kullanılmaktadır. Özellikle dijital oyunlar sağlık alanında hem tedaviyi yapan kişilerin eğitiminde hem de tedavi alan hastaların bilgi ve farkındalık düzeyinin artırılmasında olmak üzere çift yönlü bir şekilde kullanılmaktadır. Literatürde birçok farklı hastalığın tedavisinde uygulanan tedavi yöntemlerini desteklemek amacıyla geliştirilmiş birçok farklı dijital oyun bulunmaktadır. Günümüzde ise internetin ve cep telefonlarının çok yaygın kullanımından ötürü varolan hastalıklara ek olarak sosyal medya bağımlılığı çağımızın hastalığı olarak gösterilmek-

tedir [11]. Dünya nufüsunun çoğunluğunun etkin bir biçimde kullandığı sosyal medya uygulamaları birçok kişide bağımlılık yaratarak hem fiziksel hem de psikolojik rahatsızlıklara yol açmaktadır. Özellikle genç nüfusun çok sık bir şekilde kullandığı sosyal medya uygulamalarının yarattığı olumsuz etkiler bu kişilerde derin izler bırakarak birçoğunun geleceğini olumsuz yönde etkilemektedir. Bu tip istenmeyen durumları engellemek için bu çalışmada, kişilerin sosyal medya bağımlılıklarını azaltmayı hedefleyen bir ciddi oyun olan *Anti-Social* isimli oyun geliştirilmiştir. Literatürde yapılan çalışmalara göre en çok kullanılan sosyal medya uygulamaları olan Facebook, Twitter ve Instagram uygulamalarının kullanım sıklıklarını azaltmayı hedefleyen *Anti-Social*, bu uygulamaların kullanılmadıkları her süre için oyun içerisinde oyunculara ödüller vermektedir. *Anti-Social*'ın etkinliğini anlayabilmek için oyun, yaşları 21 ile 26 arasında değişen 5 kişi ile 4 hafta boyunca test edilmiştir. Testlerden elde edilen sonuçlara göre, *Anti-Social* kişilerin sosyal medya kullanımlarını azaltabilecek etkili bir oyundur.

Kaynakça

1. Baranowski, T., Blumberg, F., Buday, R., DeSmet, A., Fiellin, L.E., Green, C.S., Kato, P.M., Lu, A.S., Maloney, A.E., et al.: Games for health for children—current status and needed research. Games for health journal 5(1), 1–12 (2016)
2. Bodnar, C.A., Anastasio, D., Enszer, J.A., Burkey, D.D.: Engineers at play: Games as teaching tools for undergraduate engineering students. Journal of engineering education 105(1), 147–200 (2016)
3. Brezovszky, B., McMullen, J., Veermans, K., Hannula-Sormunen, M.M., Rodríguez-Aflecht, G., Pongsakdi, N., Laakkonen, E., Lehtinen, E.: Effects of a mathematics game-based learning environment on primary school students' adaptive number knowledge. Computers & Education 128, 63–74 (2019)
4. Brooks, S.: Does personal social media usage affect efficiency and well-being? Computers in Human Behavior 46, 26–37 (2015)
5. Chaffey, D., Ellis-Chadwick, F.: Digital marketing. Pearson UK (2019)
6. Chang, R.C., Yang, C.Y.: Developing a mobile app for game-based learning in middle school mathematics course. In: 2016 International Conference on Applied System Innovation (ICASI). pp. 1–2. IEEE (2016)
7. Chiang, I.T., Tsai, J.C., Chen, S.T.: Using xbox 360 kinect games on enhancing visual performance skills on institutionalized older adults with wheelchairs. In: 2012 IEEE Fourth International Conference On Digital Game And Intelligent Toy Enhanced Learning. pp. 263–267. IEEE (2012)
8. Chin, J.C., Tsuei, M.: A multi-modal digital game-based learning environment for hospitalized children with chronic illnesses. Journal of Educational Technology & Society 17(4), 366–378 (2014)
9. Danner, U.N., Aarts, H., de Vries, N.K.: Habit vs. intention in the prediction of future behaviour: The role of frequency, context stability and mental accessibility of past behaviour. British Journal of Social Psychology 47(2), 245–265 (2008)
10. Dickey, M.D.: K-12 teachers encounter digital games: a qualitative investigation of teachers' perceptions of the potential of digital games for k-12 education. Interactive Learning Environments 23(4), 485–495 (2015)
11. van den Eijnden, R.J., Lemmens, J.S., Valkenburg, P.M.: The social media disorder scale. Computers in Human Behavior 61, 478–487 (2016)

12. Elhai, J.D., Dvorak, R.D., Levine, J.C., Hall, B.J.: Problematic smartphone use: A conceptual overview and systematic review of relations with anxiety and depression psychopathology. Journal of affective disorders 207, 251–259 (2017)
13. Giunchiglia, F., Zeni, M., Gobbi, E., Bignotti, E., Bison, I.: Mobile social media usage and academic performance. Computers in Human Behavior 82, 177–185 (2018)
14. Gulec, U., Yilmaz, M.: A serious game for improving the decision making skills and knowledge levels of turkish football referees according to the laws of the game. SpringerPlus 5(1), 622 (2016)
15. Hainey, T., Connolly, T.M., Boyle, E.A., Wilson, A., Razak, A.: A systematic literature review of games-based learning empirical evidence in primary education. Computers & Education 102, 202–223 (2016)
16. Homer, B.D., Plass, J.L., Raffaele, C., Ober, T.M., Ali, A.: Improving high school students' executive functions through digital game play. Computers & Education 117, 50–58 (2018)
17. Kato, P.M., Cole, S.W., Bradlyn, A.S., Pollock, B.H.: A video game improves behavioral outcomes in adolescents and young adults with cancer: a randomized trial. Pediatrics 122(2), e305–e317 (2008)
18. Kwak, H., Lee, C., Park, H., Moon, S.: What is twitter, a social network or a news media? In: Proceedings of the 19th international conference on World wide web. pp. 591–600. AcM (2010)
19. Nazligul, M.D., Yilmaz, M., Gulec, U., Yilmaz, A., Isler, V., O'Connor, R., Gozcu, M.A., Clarke, P.: Interactive three-dimensional virtual environment to reduce the public speaking anxiety levels of novice software engineers. IET Software 13(2), 152–158 (2018)
20. O'Keeffe, G.S., Clarke-Pearson, K., et al.: The impact of social media on children, adolescents, and families. Pediatrics 127(4), 800–804 (2011)
21. Perrin, A.: Social media usage: 2005-2015. Washington, D.C.: Pew Internet & American Life Project (2015)
22. Primack, B.A., Shensa, A., Escobar-Viera, C.G., Barrett, E.L., Sidani, J.E., Colditz, J.B., James, A.E.: Use of multiple social media platforms and symptoms of depression and anxiety: A nationally-representative study among us young adults. Computers in human behavior 69, 1–9 (2017)
23. Raacke, J., Bonds-Raacke, J.: Myspace and facebook: Applying the uses and gratifications theory to exploring friend-networking sites. Cyberpsychology & behavior 11(2), 169–174 (2008)
24. Sampasa-Kanyinga, H., Lewis, R.F.: Frequent use of social networking sites is associated with poor psychological functioning among children and adolescents. Cyberpsychology, Behavior, and Social Networking 18(7), 380–385 (2015)
25. Shan, Z., Deng, G., Li, J., Li, Y., Zhang, Y., Zhao, Q.: Correlational analysis of neck/shoulder pain and low back pain with the use of digital products, physical activity and psychological status among adolescents in shanghai. Plos one 8(10), e78109 (2013)
26. Smith, A., Anderson, M.: Social media use in 2018. Pew Research Center 1 (2018)
27. Wakefield, L.T., Wakefield, R.L.: Anxiety and ephemeral social media use in negative ewom creation. Journal of Interactive Marketing 41, 44–59 (2018)
28. Zielhorst, T., van den Brule, D., Visch, V., Melles, M., van Tienhoven, S., Sinkbaek, H., Schrieken, B., Tan, E.S.H., Lange, A.: Using a digital game for training desirable behavior in cognitive–behavioral therapy of burnout syndrome: a controlled study. Cyberpsychology, Behavior, and Social Networking 18(2), 101–111 (2015)

Medikal Uygulama Geliştirme (MUGE) Platformu

Hande Sağlam [1], Dr. Deniz Katırcıoğlu Öztürk [1], Nihal Sarmaşık [1]

[1] ICterra Bilgi ve İletişim Teknolojileri A.Ş.
Galyum Blok Kat:2, No:3 ODTÜ-Teknokent
06531
Ankara / Türkiye

{hande.saglam, deniz.katircioglu, nihal.sarmasik}
@icterra.com

Özet. Sağlık alanında dijital dönüşüme uyumlu platform/servis/cihaz geliştirmek için dikkate alınması gereken çok sayıda parametre vardır. Güvenlik ve operasyonellik bunların başında gelmektedir. Bu tür platform, servis veya cihazların, insan sağlığı ve çevre üzerinde hayati etkileri olacağı için, hataya sebebiyet verme ve operasyon dışı kalma olasılıklarının olabilecek en düşük seviyeye getirilmesi öncelikli, hatta zorunludur. Bu zorunluluktan dolayı, altyapılar için üretilen yazılımlar, son kullanıcının genel veya hususi kullanımına açılmadan önce çok sıkı denetimlerden geçmektedir. Bu denetimlerde, sağlık alanında takip edilmesi gereken kullanıcı arayüzü standartları, pazardaki mevcut ürünlerle birlikte çalışabilirliği sağlayacak iletişim standartlarına uyumluluk aranmaktadır. Dijital dönüşüme uyumlu sağlık aktörlerinin olmazsa olmazı olan birlikte çalışabilirlik esasının, hataya sebebiyet vermeme, etkin ve doğru çalışma, veri güvenliğini garantileme, geriye dönük kanıt sağlayabilme, operasyonel ve iletişimde kalabilme vb. gibi geniş bir spektruma izdüşümü bulunmaktadır. Dijitalleşmenin ve kanıta dayalı tıp esasına etkin girdi sağlayabilmenin bir sonucu olarak birbirleriyle yoğun iletişimde olması gereken, yani, birlikte çalışabilirlik ilkesini sağlaması beklenen bu yazılımlar, ortak fonksiyonlar barındırmaktadır. Bu motivasyonla, söz konusu bağlamdaki yazılımların geliştiricilerine, hız kazandıracak ve eforlarını doğru odaklayarak verecekleri katkıyı maksimuma çıkaracak ve geliştirme süreçlerine denetimli bağlam sınırı getirecek mimari bir çatı önerilmektedir. Bu çatı ile, farklı platformların geliştirme süreçlerinde kanıtlanmış bir mimariyi sağlık bağlamıyla buluşturarak, tekrar edilecek geliştirme eforunu minimuma indirgeyen, kod yeniden kullanımını destekleyerek, hatalı birim üretme olasılığını düşüren, sıkça kullanılan katmanlı yapısı sayesinde ayrışmış, hedefli ve ek yetenekler için rahatlıkla genişleyebilen bir geliştirme platformu sağlanması kurgulanmıştır.

Anahtar Kelimeler: Sağlık, Medikal Cihaz, Medikal Uygulama, e-Sağlık, Altyapı

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Medical Application Development (MADE) Platform

Hande Sağlam [1], Dr. Deniz Katırcıoğlu Öztürk [1], Nihal Sarmaşık [1]

[1] ICterra Information and Communication Technologies
Galyum Blok Kat:2, No:3 ODTÜ-Teknokent
06531
Ankara / Turkey

{hande.saglam, deniz.katircioglu, nihal.sarmasik}
@icterra.com

Abstract. While developing a platform/service/device which is suitable for digital healthcare infrasturcutures, there exist many parameters that developers must consider. Safety and functionality are the parameters coming forefront. It is not acceptable that these platforms/services/devices to be error-prone or dysfunctional due to the fact that they do have vital effects on human life and environment. To avoid any malfunction, the software developed for these frameworks are strictly analyzed and tested upon consumer use. Compatibility to international standards of interoperability is heavily scrutinized. There is a projection of a wide spectrum on the interoperability principles of digital actors such as failure to cause error, effective and correct operation, guaranteeing data security, providing backward evidence, and staying in operational and communication. The software developed for interoperable systems have to be tightly bound to each other as a result of digitization and evidence-based medicine criteria. It is been recommended for developers in this domain to use a proven software architecture framework to fasten the development process, while maximizing the contribution with focused effort and enabling initiatives to stay in the healthcare scope. With the suggested model, MADE, it is aimed to minimize the redundant development efforts by combining a proven architecture with the health context, to reduce the likelihood of producing faulty componets by supporting code reuse and to provide a development platform that is targeted and extensible for auxillary capabilities.

Keywords: Health, Medical Device, Medical Application, e-Health, Framework

1 Giriş

Bakım, donanım yenileme ve onarım süreçleri, sarf malzeme ihtiyacı, yazılım güncelleme masrafları ve sürekli değişen, yenilenen regülasyonlar ile birlikte medikal bağlamın paydaşlarına doğrudan ve dolaylı yoldan maliyet yükü artmaktadır. Sektörün mevcut durumu ve geleceğine yönelik analizler, dünya piyasalarına yön veren teknolojiler ve Sağlık Bakanlığı'nın uzun vadeli planları doğrultusunda, yerli kaynaklarla ileri teknoloji içeren ve katma değeri yüksek medikal cihazların geliştirilmesi çalışması hız kazanmıştır. Bu çalışmalarda medikal görüntüleme cihazları önemli yer tutmaktadır. Gelişen teknoloji ve kullanıcı deneyimleri göz önüne alındığında medikal görüntüleme cihazlarının konsol yazılımları, ürünün muadillerinden ayrışmasında oldukça belirleyici olmaktadır. Medikal uygulamalarda veri bütünlüğünün ve gizliliğinin önemli olması ve uygulamaların çoğu zaman gerçek zamanlı çalışma ihtiyacı konsol uygulamalarının masaüstü uygulama olarak geliştirilmesini kaçınılmaz kılmaktadır.

Emniyet kritik sistemler hatalı çalışması durumunda insan hayatı kaybına sebep olan, çevreye veya mal/mülke zarar veren sistemlerdir. Bu sistemler için geliştirilen yazılımlar ise emniyet kritik yazılım olarak adlandırılır ve sağlık alanında geliştirilen uygulamaların neredeyse tamamı bu sınıfa dahildir. İnsan hayatı faktörü devrede olunca, geliştirme ve üretim süreçleri standartlarla sınırlanmakta ve düzenlenmektedir. Bu regülasyonlara uyum sağlamak için de, başarısı kanıtlamış referans mimariler kullanarak geliştirme yapmak uygulamanın oluşturabileceği hataların ortaya çıkma riskini epeyce azaltacaktır.

Medikal cihazlar için diğer bir önemli faktör ise, birlikte çalışabilirliktir. Çünkü sağlık yazılım sistemleri bütünüyle çok büyük bir ailedir ve genel olarak bu ailedeki her bir birey birbiriyle direk ya da dolaylı olarak haberleşme ihtiyacı duyabilir. Birlikte çalışabilirliği sağlayabilmek için de, yazılım yaşam döngüsünde olduğu gibi, tanımlı standartların takip edilmesi gerekmektedir. Bu standartlar oldukça kapsamlı olup, cihazların genel davranışlarını ve birbirleri ile haberleşme yöntemlerini tanımlar. Bu standartlara göre görüntü (dalga biçimli, zaman serisi, 2D, 3D, 4D vb.) üretebilen tıbbi cihazlara *modalite* adı verilir. Bu bağlamda elde edilen bilgi birikimi, tanımlı iletişim arayüzleri ve gelişkin işleme algoritmaları benzer özelliklerde geliştirilen diğer modalitelerde de etkin şekilde kullanılabilir.

Sağlık alanında uygulama ya da cihaz geliştiren küresel aktörlerin stratejilerine bakacak olursak, tek cihaz ya da uygulama geliştirmekten ziyade komple çözüm sunmak üzerine yoğunlaşıldığı fark edilecektir. Bu nedenle de, tescilli altyapılar hazırlayıp yazılımların devamlılığını ve kolay üretilebilirliğini sağlamışlardır. Bu altyapılar, ticari sır kapsamında oldukları için piyasada genel kullanıma ya da satın alıma açık değildirler.

Bu şartlar bir bütün olarak düşünüldüğünde, şirketimiz bünyesinde **M**edikal **U**ygulama **GE**liştirme (**MUGE**) platformu adı verilen bir uygulama geliştirilmesi yönünde adım atılmış ve ilgili kapsam TÜBİTAK TEYDEB 1501 Sanayi AR-GE programı altında 3180586 no'lu proje ile desteklenmeye esas hale gelmiştir.

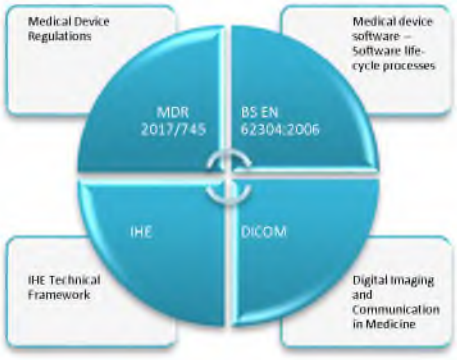
2 Motivasyon

Sağlık alanında geliştirilen uygulamaların birlikte çalışabilirlik, hataya sebebiyet vermeme, etkin ve doğru çalışma, veri güvenliğini garantileme, geriye dönük kanıt sağlayabilme, operasyonel ve iletişimde kalabilme vb. gibi olmazsa olmaz zorunlulukları vardır. Bu nedenle, bu bağlamdaki yazılımların geliştiricilerine, hız kazandıracak, eforlarını doğru odaklayarak verecekleri katkıyı maksimuma çıkaracak ve geliştirme süreçlerine denetimli bağlam sınırı getirecek mimari bir çatı önerilmektedir. Bu ihtiyacı gidermek için önerilen MUGE uygulaması kapsamında, öncelikli olarak bir medikal görüntüleme cihazı uygulamasının gerçeklenmesi için gereksinimler analiz edilmiş ve bu doğrultuda, sağlık alanında geliştirilebilecek başka uygulamalar da mümkün mertebe göz önünde bulundurularak, ortak uygulama çatısı mimarisi tasarlanmaya çalışılmıştır.

Uygulama çatısı kullanılmadan geliştirilen projelerin daha sonraki projelerde kullanılması ya mümkün olmamaktadır ya da yalnızca kod seviyesi yeniden kullanımlar mümkün olabilmektedir. Daha büyük bileşenler seviyesinde yeniden kullanımın sağlanmasında uygulama geliştirme çatıları büyük önem taşımaktadır. Bazı çatıların kullanımı zordur ve ciddi uzmanlık gerektirmektedir. Geliştiricilerin işlerini basitleştirmek için bu tarz karmaşık çatıların sadeleştirilerek ve basit bir çatı altında sarmalanarak sunulmasının, geliştirme eforunu ve yetkinlik eksikliği sebebi ile yapılacak hatalı geliştirme durumlarını önemli ölçüde azaltacağı düşünülmektedir. Morisio ve arkadaşlarının yaptıkları çalışmada, bir mimari çatı kullanılarak geliştirilen yazılımların, kullanılmayanlara göre daha kısa sürede, daha kolay ve az kod yazarak geliştirildiği ve yeniden işleme eforunun daha az olduğu görülmüştür [1]. Schmidt ve arkadaşlarının yaptığı bir diğer çalışmada ise, genişleyebilir, esnek, taşınabilir, güvenilir, ölçeklenebilir ve maliyetine katlanılabilir bir yazılım için çatı kullanımının önemi vurgulanmaktadır. Bu çalışmada ayrıca bir çatının sahip olması gereken karakteristik özellikler ve yeni bir çatı geliştirmeye karar verildiğinde göz önünde bulundurulması gereken noktalar ayrıca belirtilmektedir [2].

Uygulama çatısı ve başarısı kanıtlanmış mimari tasarım öğelerinin projelerde kullanımı, ileride karşılaşılabilecek olası gereksinim/ihtiyaç değişikliklerine de kolay adapte olabilmeyi sağlayacaktır. Bu altyapı ile geliştirilmiş olan uygulama, modüler olacaktır ve gereksinim değişiklikleri nispeten kolay yönetilebilecektir. Bilindiği üzere, projelerin geliştirme aşamalarında ne kadar ilerlenmiş olunursa, gereksinim değişikleri o denli maliyetli olmaktadır. Gupta ve arkadaşları, yaptıkları çalışmada yeniden kullanılabilir bileşenler ile yeniden kullanılabilir olmayan bileşenlerde yapılan değişikliklerin karakteristiklerini incelemişlerdir. Bu çalışmada, zaman baskısı, olgun ve kararlı olmayan gereksinimler nedeni ile çoğu yazılım projesinde ilk seferde optimal çözümün yakalanamadığı ve bu nedenle daha sık kod seviyesi düzenleme yapıldığı vurgulanmıştır [3].

Sağlık alanında geliştirilecek uygulamaların önemli bir alt grubu için emniyet kritiklik ve birlikte çalışabilirlik elzemdir ve gerçeklenebilmesi için uluslararası standartlara uyulması zorunludur. Tıbbi görüntüleme cihazları için bu standartlar Şekil 1'de özet olarak gösterilmiştir.



Şekil 1. Medikal cihaz geliştirme standartları

MDR 2017/745 Medical Device Regulations

Tıbbi cihaz yönetmeliğidir ve Avrupa Birliği konseyi (EC) tarafından oluşturulmuştur [4]. Sağlık Bakanlığı tarafından ülkemize uyumlandırılarak yürürlüğe alınmıştır [5]. Bu yönetmelik, insanların sağlığı için kullanılan aktif tıbbi cihazlarda bulunması gerekli temel özelliklerin belirlenmesine, bu cihazların tasarımı, üretimi ve tüketicilere sunulmasına, tüketiciler tarafından kullanılmasına, kullanımı sırasında hastaların, uygulayan görevlilerin, kullanıcıların ve üçüncü kişilerin sağlık ve güvenliği açısından tehlike yaşamamalarını sağlayacak şekilde piyasaya sunulmasına, ve bu cihazların denetlenmesine yönelik esasları düzenlemektir.

BS EN 62304:2006 Medical device software -- Software life-cycle processes

Yazılım içeren tıbbi bir cihazın emniyetinin ve etkinliğinin sağlanması, yazılımın işlevini kabul edilemez risklere neden olmadan yerine getirdiğinin gösterilmesini gerektirir. Bu standart, tıbbi cihaz yazılımının emniyetli tasarımı ve bakımı için gerekli aktiviteleri bir yaşam döngüsü çerçevesinde tanımlamaktadır.

Digital Imaging and Communication in Medicine (DICOM) Standart

DICOM [6], medikal uygulamaların birlikte çalışmasını sağlamak için belirlenmiş bir standarttır ve uluslararası kabul gören ortak bir iletişim dili olarak değerlendirilmektedir. Cihazların ya da uygulamaların birbiri ile iletişimde olabilmesi için haberleşme ve mesajlaşma protokolleri, komutların sözdizimi, semantiği ve bu protokoller kullanılarak gönderilen bilgileri belirler. Bu standart herhangi bir cihaz veya uygulama türüne özelleşmiş değildir. Sağlık alanında yer bulan tüm cihaz gruplarını kapsar ve standardı uygulayacak merci ürettiği/geliştirdiği modaliteye uygun olanı seçip yorumlamakla sorumludur. Ciddi bir analiz ve öğrenme süreci gerektiren bir standarttır. Cihazların veya uygulamaların birlikte çalışabilirliği için takip edilmesi gereken ve uyumluluk aranan bir standarttır. DICOM bu ihtiyacı karşılayan tek standart değildir, aynı ihtiyacı karşılayacak başka standartlar da vardır ancak yaygınlığı sebebiyle DICOM kullanılması öngörülmüştür.

Integrating the Healthcare Enterprise

Integrating the Healthcare Enterprise (IHE) [7], hastane bilgi yönetim sistemlerinin birlikte çalışabilirliğini ve elektronik medikal kayıtlarının etkin kullanımını sağlamak için standartların kullanımını destekleyen bir girişimdir. IHE, çeşitli klinik ve operasyonel

alanlardaki hizmet sağlayıcılarının gönüllü komisyonlarının kritik birlikte çalışabilirlik konularında standartlara dayalı çözümler konusunda fikir birliğine varmaları için bir forum sunmaktadır. IHE var olan standartları kullanarak klinik gereksinimlere çözüm üretir, bir standart tanımlamaz. Genel olarak iş akışları belirler ve Radyoloji terapötik alanı temelinde DICOM standardına referans verir. DICOM standardı daha alt seviye detaylar içerir, IHE ise DICOM kullanarak cihazlara ya da uygulamalara alacakları roller doğrultusunda hangi DICOM servislerinin geliştirilmesi gerektiğini, hangi ek yeteneklerin sağlanması gerektiğini tanımlar. Yalnızca, klinik alanlardaki işlemleri tanımlamakla kalmaz; operasyonel, altyapısal roller (bilgi işlem altyapısı) ve iş akışlarını da bünyesinde barındırır.

Medikal alanda geliştirilecek uygulamalar ya da görüntüleme cihazları için onaylı kuruluş ve yetkin merciler IHE ve DICOM uyumluluğu istemektedir. Bunun temel gerekçesi, yeni geliştirilen bir ürünün piyasaya etkin ve doğru sunumu sırasında entegrasyon sorununun yaşanmamasıdır. Onaylı kuruluşlar tarafından sağlanan uyumluluk belgesini alabilmek için ise, dünya çapında *Connectathon* [8] adı verilen etkinlikler düzenlenmektedir. Bu etkinliğe geliştirilen ürün ve katılımcı olarak katılmak için önceden bir test sürecinden başarılı olarak değerlendirilmek gerekmektedir. Connectathon etkinliğine katılımcı olmak için hangi IHE profilleri hangi rol kapsamında geliştirildi ise, ilgili profile ait tanımlanmış ön test prosedürünün uygulanıp, sonuçlarının komiteye gönderilmesi beklenir. Bu ön değerlendirme sonucunda geliştirilen ürün yeterli görülmezse, bu etkinliğe kabul edilmez. Yeterli olarak değerlendirildiği durumda ise, etkinlikte yeterli görülen diğer cihazlarla dijital iletişim kurarak beklenen doğrulama prosedürlerinin uygulanması gerekir. Cihaz ilgili IHE profilinin ilgili aktörü rolünde olarak, aynı profilin başka aktörleri ile doğrulama sağlar ise, o profil için uygunluk alabilir hale gelmiş demektir [9].

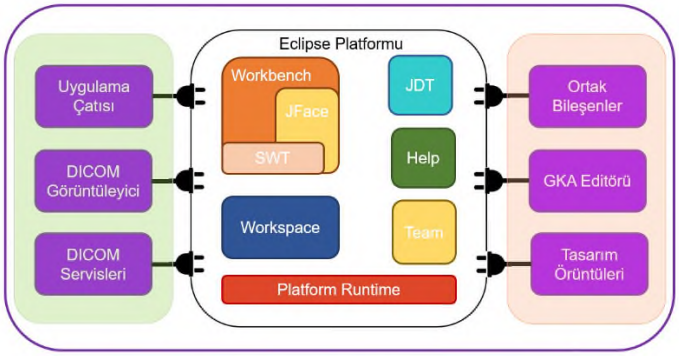
Türkiye'de de Sağlık Bakanlığı, Sağlık Turizmi süreçlerinin ve uzaktan tıp hizmetlerinin hastaneler tarafından etkin ve şeffaf yönetimi için Teletıp altyapılarının kurgulanarak uygulanmasını zorunlu tutmuştur. Hatta, bu amaçla Sağlık Bakanlığı'na bağlı Teletıp ve Teleradyoloji Birimi kurulmuştur. Yayımladıkları yönergede, bu birimin görev ve sorumluluklarını şu şekilde sıralamışlardır.

- Teleradyoloji standartlarını araştırmak, geliştirmek ve ilgili sistemlere entegrasyon çalışmalarını yürütmek.
- Teleradyoloji alanında "birlikte çalışabilirlik" uygulamalarını iyileştirmek ve paydaşlarla işbirliği yapmak.
- Birlikte çalışabilirlik esaslarını benimseyerek Teleradyoloji standartlarının çalışma alanlarına uygulanmasını sağlamak [10].

Sonuç olarak, Türkiye pazarında geliştirilecek sağlık uygulamalarında teknoloji son derece yakından takip edilip, uluslararası standartları yakalar duruma getirilmiştir. Bu nedenle geliştirilecek uygulamalardaki IHE ve DICOM uyumluluğu olmazsa olmaz bir hale gelmiştir.

3 Kapsam

Bu bağlamda, öncelikli olarak tıbbi görüntüleme cihazı uygulamalarındaki geliştirmeyi kolaylaştırmak amacıyla bir platform ihtiyacına odaklanılmıştır. Bu haliyle MUGE, geliştiricilerinin ihtiyaç duyduğu araçları, arayüzleri ve servisleri bir arada sunan Eclipse [11] tabanlı özelleşmiş bir platform oluşturmayı hedeflemektedir. Eclipse, modüler mimarisi sayesinde bileşen tabanlı geliştirme olanağı sunmaktadır. MUGE platformu kapsamında da, Eclipse'in bu yeteneğinden faydalanılarak Şekil 2'deki gibi bir mimari tasarım kurgulanmıştır. Bu esnek mimariyle, ortaya çıkan platformu hem medikal alanda hem de medikal olmayan alanlarda kullanmak mümkün olacaktır.



Şekil 2. MUGE mimari tasarımı

MUGE bileşenleri yeni proje oluşturma aşamasında geliştiricinin ihtiyaçları doğrultusunda tak-çalıştır modeliyle eklenebilecektir. Bir diğer deyişle, parametrik olarak yalnızca gerekli bileşenler eklenerek proje oluşturulabilecektir.

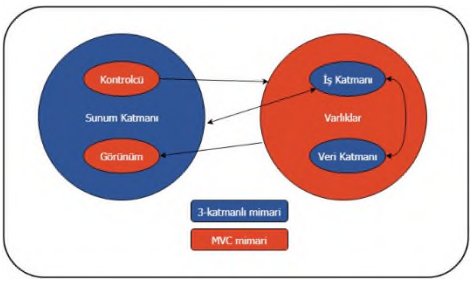
MUGE projesi kapsamında geliştirilecek ürünün kullanıcıları son tahlilde yazılım geliştiricileri olacaktır. Bu nedenle, geliştirilecek her bir bileşen için başarı hedefi yazılım geliştiricinin işinin ne denli kolaylaştığı, kod yeniden kullanımlarını arttırarak, geliştirme maliyetini ne denli azalttığı ve kararlı altyapıların geliştirme sürecinde ne denli yer bulduğu ile eşgüdümlüdür. Sağlık alanına özel olarak, bu tarz bir altyapı yardımı ile uygulama geliştirmek, ortaya çıkan ürünün daha kararlı olmasını, ve ürünün zorunlu standartlara uygunluğunun tarafsız şekilde denetlenebilmesini sağlayacaktır.

Geliştirilecek bileşenlerin kısa özeti ve hedefleri şu şekilde incelenebilir.

Uygulama Geliştirme Çatısı, MUGE aracılığı ile geliştirilecek uygulamaların belirli mimari kalıplar içerisinde geliştirilmesini sağlayacaktır. MUGE kullanıcısının çalışacağı projeyi, yeni proje yaratma sihirbazı aracılığı ile ihtiyaçlarına yönelik olarak kolayca oluşturması hedeflenmektedir. Öncelikli olarak 3-katmanlı mimari (Sunum-İş-Veri) paketleri ve bağımlılıkları projeye eklenecektir. 3-katmanlı mimari, en yaygın mimari tasarım modellerinden biridir. Bu mimari tasarım modelinin bilinen en kritik dezavantajı, katmanların ve bu katmanlar arasındaki ilişkilerin kurulmasıdır [12]. MUGE ile hazırlanan projelerde bu yük ortadan kalkmış olacaktır. Her platformda

yaygın hakimiyeti olan MVC (model-view-controller) tasarım örüntüsü ise, 3-katmanlı mimari tasarım modeli ile uyum içerisinde çalışmaktadır.

Şekil 3'de 3-katmanlı mimari ve MVC tasarım örüntüsünün birlikteliği görülmektedir. Sunum katmanında "Görünüm (View)" ve "Kontrolcü (Controller)" sınıfları tanımlanacak ve bu sınıflarının iş mantığı gereği kullanacağı "Varlıklar (Model)" ile ilişkisi tanımlanacaktır. Bu noktada, "Inversion-of-Control" [13] mekanizması ile birbirleri ile olan mesajlaşma altyapısı hazır olarak sunulacaktır.



Şekil 3. MVC ile 3-katmanlı mimarinin birlikte çalışma modeli

Genel bağlamda uygulama geliştirilirken, ihtiyaçlara yönelik farklı bir çok dış kütüphaneye bağımlılıklar oluşabilmektedir. Bu bağımlılıkların yönetimi de, projeler için ayrıca ele alınması gereken bir konu olarak karşımıza çıkmaktadır. Hem kendi proje hiyerarşisi içindeki modüllerin bağımlılıklarını, hem de üçüncü parti kütüphanelere olan bağımlılıkları ortak bir yerden kolayca yönetmek için bu kapsamda önerilen altyapı, bağımlılık yönetimi sürecini kolaylaştıracaktır.

Görüntüleme Bileşeni, MUGE ile geliştirilecek tıbbi görüntüleme cihazı odaklı uygulamaların gelişkin görüntüleme desteği sunması geliştiricinin işini epeyce kolaylaştıracaktır. Bu bileşen, özellikle farklı boyutlarda çıktı sağlayan görüntüleme ve biyosinyal monitör etme cihazları için geliştirilecek uygulamalarda anlamlı olacaktır. Bu bileşenin, endüstri standardı olarak yaygın şekilde kullanılan DICOM formatını desteklemesi ve gelişmiş görüntüleyici özelliğine sahip olması hedeflenmektedir. Öncelikli olarak Dijital Radyografi (DX) modalitesi ile üretilen 2 boyutlu (2D) görüntülerin gösterimi üzerine özelleşilecektir. Bu görüntü bilgisinin tamamen DICOM veri formatından okunup, bir görüntüleyici girdisi olarak kullanılması hedeflenmiştir. DICOM veri formatı detay bilgisi veya ilişkili geliştirme altyapısına etraflıca hakim olmayan bir geliştirici için de, görüntüleyici entegrasyonu olabildiğince basit hale getirilmiş olacaktır. DICOM standardı tıbbi bağlama özgü bir standart olduğu için ciddi bir öğrenme ve özümseme süreci gerektirmektedir. Aynı şekilde, görüntü üzerinde yapılması ön görülen ve temel görüntü işleme uzmanlığı gerektiren operasyonların da geliştirici için hazır kullanılabilir halde sunulması hedeflenmektedir. Ancak ilerleyen aşamalarda, sadece 2 boyutlu verinin değil, 1, 2 ve 3 boyutlu tüm DICOM uyumlu klinik veri setleri için de kapsamlı bir işlevsellik eklenmesi düşünülmektedir. Bu bileşen ile ilgili varılmak istenen en iyi nokta ise, bağımsız olarak kullanılabilir ve tüm platformlara (web/mobil/masaüstü) entegre edilebilir bir modelde çalışmasıdır.

Tıbbi görüntüleyici cihaz uygulamalarında yeni trend dokunmatik ekran desteğidir. Bunun sağlık alanına özel sebebi, perifer olarak klavye-fare kullanımının ya da ekran etrafında yerleştirilen kullanıcı etkileşimli butonların mümkün mertebe elimine edilerek, tıbbi süreçlerin işletilmesine mani olmaması ve bu sayede hatırı sayılır düzeyde performans artışı sağlamış olmasıdır [14]. Buradan yola çıkılarak, kullanıcı arayüzlerinin dokunmatik olarak çalışmasını destekleyen çoklu dokunma (multi-gesture) altyapısı hazır olarak sunulacaktır.

Kullanıcı Arayüzü Geliştirme Bileşeni, MUGE ile geliştirilecek projeler için kritik öneme sahiptir. Bu bileşen, yalnızca, sağlık uygulamalarını özel olarak hedeflemekle kalmayarak, tüm önyüz etkileşimli form tasarımı içeren masaüstü uygulamaları için kullanılabilir olacaktır. Yazılımcılar için arayüz tasarlamak ya da tasarlanan arayüzü gerçeklemek yaygın olarak maliyetli bir iş olarak değerlendirilir. Bu yükü hafifletmek adına, söz konusu bileşenin bünyesinde yer alacak Sürükle-Bırak özellikli editör desteğinin, Swing ve JavaFX ile tanımlanmış olan kullanıcı arayüz bileşenleri yardımıyla geliştiricinin işini kolaylaştırması beklenmektedir. Öncelikli olarak Java dili desteği hedeflendiği için, arayüz kütüphanesinde Swing ve JavaFX bileşenlerinin yer alacağı arayüz geliştirme araç çubuğu eklenecektir. Sağlık alanında geliştirilecek uygulamalardaki adaptasyon ve standardizasyon için öğrenme sürelerini minimuma indirmek maksadıyla, ilgili bağlamın kullandığı ve yer yer zorladığı arayüz geliştirme standartlarına [15] uyulması sağlanacaktır. Böylece geliştiricinin, tıbbi arayüz geliştirme hususunda uzmanlık sahibi olmadan sonuç üretebilmesi mümkün olacaktır. Swing ve JavaFX temel bileşenleri dışında yaygın kullanım alanı bulabilecek kompleks bileşenler (modüler liste, tablo vb.) de sunulacaktır. JavaFX kütüphanesi bilhassa çoklu dokunma yetenekleri sebebiyle kullanılacaktır. Swing ve JavaFX altyapılarının bir araya getirilmesi geliştiricilerin genellikle çekince duyduğu bir manuel bir entegrasyona karşılık gelmektedir ve MUGE kapsamında bu birlikteliğin hazır olarak sunuluyor olması geliştirme sürecinde büyük hız sağlayacaktır.

DICOM Servisler Bileşeni & IHE İş Akışı Bileşeni, MUGE bileşenleri arasında sadece sağlık alanına hitap eden ve belirli modaliteler kapsamında geliştirilecek projeler için anlamlı olan bir modüldür. Bir modalite üzerinde çalışacak uygulama, mevcut ekosistemdeki diğer ilişkili uygulamalar ile birlikte çalışabilir olmalı ise, DICOM standardını ve IHE profilleri ile tanımlı iş akışlarını gözetiyor olmalıdır. Bu birlikte çalışılabilirlik esasına uygunluğu bir çok sağlık sistemi yönergesi zorunlu tutmaktadır. Özellikle, T.C. Sağlık Bakanlığı Proje Geliştirme Daire Başkanlığı yayınladığı bildiride [16] DICOM standardı entegrasyonunu mecbur tuttuğunu belirtmiştir. MUGE kapsamında, bu beklentilerle eşgüdümlü olarak, geliştiriciler tarafından modaliteye özel seçilen IHE iş akışları ve bu akışlar doğrultusunda geliştirilmesi zorunlu olan DICOM standardıyla tanımlı servislerin otomatik olarak proje ortamına eklenebilmesi kurgulanmıştır. İlk etapta, daha küçük bir iş akış ailesi desteği eklenecek, sonrasında ise desteklenebilecek modalite çeşitliliği yönünde ek yetenekler dahil edilecektir.

Tasarım Örüntüleri Bileşeni, MUGE ile geliştirilecek projelere eklenerek projelerin bu mimari özellikleri kolaylıkla desteklemesi sağlanacaktır. Tasarım örüntüleri, yazılım tasarımında yaygın olarak ortaya çıkan bir soruna karşı oluşturulmuş olan genel tekrarlanabilir bir çözümdür. Bir tasarım örüntüsü, doğrudan koda dönüştürülebilen bitmiş bir tasarım değildir. Birçok farklı durumda kullanılabilecek ve bir sorunun nasıl

çözüleceğine dair açıklamaları içeren bir şablondur. Tasarım örüntüleri, test edilmiş, kanıtlanmış paradigmaları sağlayarak geliştirme sürecini hızlandırır. Tasarım örüntülerini kullanmak, büyük sorunlara yol açabilecek adımları önlemeye yardımcı olurken, örüntüleri bilen geliştiriciler için kod okunabilirliğini arttırır. Ancak, tasarım örüntülerini tasarı aşamasında kullanmak yazılacak kod satır sayısını genellikle arttırdığı için, yazılımcılar tarafından sıklıkla ihmal edilebilmektedir. MUGE bu bileşeni aracılığıyla, geliştiriciyi tasarım örüntüsü altyapısını hazırlamaktan kurtararak, uygulamada hız kazanılmasını sağlayacaktır. Bir eklenti (plug-in) olarak sunulacak tasarım örüntüleri bileşeni, Gang of Four (GoF) [17] tarafından önerilmiş olan tasarım örüntüleri şablonları kullanılarak geliştirilecektir. Bu kapsama dahil olan tasarım örüntüleri şablonları Tablo 1'de üç kategori halinde listelenmiştir.

Tablo 1. MUGE kapsamında geliştirilecek tasarım örüntüleri listesi

Creational	Structural	Behavioral
Abstract Factory	Adapter	Chain of Responsibility
Builder	Bridge	Command
Factory Method	Composite	Interpreter
Prototype	Decorator	Iterator
Singleton	Facade	Mediator
	Flyweight	Memento
	Proxy	Observer
		State
		Strategy
		Template Method
		Visitor

Ortak Bileşenler, MUGE bileşenleri içerisinde projelerin farklı katmanlarının ihtiyaç duyabileceği, olmazsa olmaz bileşenleri içeren bir modüldür. Bu sınıflamada düşünülen fonksiyonların, tüm uygulamayı etkileyeceği göz önünde bulundurularak, kodlama hiyerarşisi bünyesinde mümkün olduğunca tek noktada birleştirilmeleri doğru pratiktir. Bu amaçla ilgi-alanı yönelimli programlama (aspect-oriented programming) yaklaşımı doğrultusunda, ortak bileşenler modüler yapıda tasarlanacak, birden fazla katmanı etkileyecek olan davranışlar tek modülde bir araya getirilerek uygulama genelinde tek noktadan erişilip kullanılması sağlanacaktır. Bu bakış açısıyla, güvenlik yönetim bileşeni, günlük bileşeni ve veri yönetim bileşeni gibi temel öncelik sahibi bileşenler bu ortak yapı içerisine dahil edilecekler arasındadır. Güvenlik yönetim bileşeni, geliştiricinin uygulamaya giriş yapan kullanıcı için kimlik denetimi ve yetkilendirme işlemlerini yapabilmesi için gerekli yetenekleri sunan altyapıdır. Günlük bileşeni, genel olarak tüm uygulamaların günlük tutma ihtiyaçlarını karşılamak için bileşendir. Veri Erişim Bileşeni ise, veritabanı kullanan uygulamalar için veriye erişim ve geliştiricinin işlemsel (transactional) fonksiyonları yapabilmesi için katmanlı altyapı sunan bileşendir.

Güvenlik ve günlük bileşeni sağlık alanında geliştirilen projeler için oldukça büyük öneme sahiptir. Sağlık süreçlerinin ve hasta verilerinin önemli role sahip olduğu bu bağlamda, denetimler, oluşan hatalar ve olası görev ihmali veya suistimali durumları

hukuki süreçlere esastır ve sistemlerin günlük bileşeni sayesinde tutulan sayısal kayıtlar kanıt niteliği taşımaktadır. Tıbbi görüntüleme cihazları ya da uygulamalarında karşılaşılan küçük/büyük tüm hata durumlarında, mevcut uygulamanın o anki durumu zaman pulu eşliğinde dondurularak, güvenli bir formatta saklanması kritik öneme sahiptir. Öte yandan, günlük bileşeni barındırıyor olmasının bir projeye en yadsınamaz katkısı, geliştirmesi tamamlanmış bir uygulamanın çalışırken oluşan beklenen/beklenmeyen hatalarının, geliştirici tarafından kolayca tespit edilmesine ve anlamlandırmasına olanak sağlamaktır. Kolayca sisteme entegre edilen bir günlük bileşeni altyapısı, geliştiricinin yükünü hafifletecek ve geliştirme aşamasında doğru kırılma noktalarını tespit ederek doğru yöntemi kolayca üretilebilmesini sağlayacaktır.

4 Sonuç

Öncelikli hedef olarak tıbbi görüntüleme cihazı uygulamaları için altyapı sağlayacak olan MUGE platformu, geliştiricilerin sağlık projelerinde tekrar tekrar yapması beklenen işlerin azaltılarak, işleyişin katma değerli noktalarına daha fazla zaman ayırmalarına olanak sağlayacaktır. Aynı zamanda, kod yeniden kullanımını arttırarak daha kararlı yazılımlar geliştirilmesini mümkün kılacaktır. Mimari tasarım altyapısı ve tasarım örüntüleri şablonları ile daha okunabilir, kolay anlaşılabilir, genişleyebilir ve modüler yapıda projeler üretilmesi olanaklı olacaktır. Geliştiricilere öğrenme maliyeti yüksek olan DICOM standardına ve IHE profillerine uyumluluğu hazır olarak getirecektir. Genel olarak tüm uygulamalarda ortak olarak bulunması beklenen bileşenler, bu platformun parametrik yapısı sayesinde kolayca entegre edilebilecektir. MUGE platformu ile üretilen yazılımlarda süreçte oluşan veri ve modelini, kanıta dayalı tıp adına daha anlamlı ve tanı/tedavi süreçlerinde daha hedefli kullanmak için entegre edilebilir arayüzler geliştirmek ulaşılması istenen önemli bir noktadır.

Kaynakça

1. Morisio, M., Romano, D., Stamelos, I.: Quality, Productivity and Learning in Framework-Based Development: an Exploratory Case Study. IEEE Transactions on Software Engineering. (2002).
2. Schmidt, D.C., Gokhale, A., Natarajan, B.: Frameworks: Why They Are Important and How to Apply Them Effectively. (2004).
3. Gupta, A., Conradi, R., Shull, F., Cruzes, D., Ackermann, C., Rønneberg, H., Landre, E.: Experience Report on the Effect of Software Development Characteristics on Change Distribution. PROFES 2008: Product-Focused Software Process Improvement. 158–173 (2008).
4. REGULATIONS. 117, (2017).
5. Çevirisi, AB Mevzuat Birimi, https://titck.gov.tr/storage/announcement/MDR BÖLÜMLER.pdf.
6. https://www.dicomstandard.org/.
7. https://www.ihe.net/.
8. https://connectathon.ihe-europe.net/.
9. https://www.ihe-europe.net/sites/default/files/WP_Connectathon_2019_0.pdf.

10. Teletıp ve Teleradyoloji Sistemi. (2018).
11. https://www.eclipse.org/.
12. Sarma, S.: 3-Tier Architecture. (2010).
13. https://en.wikipedia.org/wiki/Inversion_of_control.
14. Hawkins, E.: https://www.ecnmag.com/article/2013/06/touch-screens-medical-instruments.
15. Medical devices -- Application of usability engineering to medical devices. (2015).
16. https://sbsgm.saglik.gov.tr/TR,12844/teletip-ve-teleradyoloji-birimi.html.
17. Gamma, E., Vlissides, J., Johnson, R., Helm, R.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley (2009).

Sağlık Bilgi Sistemi ve Teknolojisi Olgunluk Faktörleri

Samet Can[1], Ayça Tarhan[2]

[1,2]Bilgisayar Mühendisliği, Hacettepe Üniversitesi, 06800, Ankara, TÜRKİYE
[1]samet.can@hacettepe.edu.tr, [2]atarhan@hacettepe.edu.tr

Özet. Dinamik, değişken ve çok-disiplinli süreçlerin hâkim olduğu sağlık alanında hizmet süreçlerinin kalitesi ve verimliliği, bu süreçleri destekleyen bilgi sistemi ve teknolojisi (BST) olgunluğundan doğrudan etkilenmektedir. Literatürde sağlık BST olgunluğunu değerlendirmek veya güvence etmek için önerilmiş çok sayıda model yer alsa da yaygın kabul görmüş bir model bulunmamaktadır. Bu bildiride, bahsedilen çalışmalarda münferit olarak önerilmiş 48 adet Sağlık BST Olgunluk Modelinden hareketle, sağlık alanında BST olgunluğunu değerlendirmeye yönelik faktörler çıkarılmış ve Temellendirilmiş Kuram metodolojisiyle sınıflandırılmıştır. Bu bildiri ile önerilen sınıflandırma ve olgunluk faktörleri, sağlık BST olgunluğu üzerine çalışan araştırmacılara ve pratisyenlere, yapacakları deneysel çalışmalarda referans olabilir. Gelecek çalışmalarda, bu bildiride sunulan faktörlerin, sağlık kurumlarının BST olgunluğunu anlamak için yapılacak bir ankete temel olarak kullanılması planlanmaktadır.

Anahtar Kelimeler: Sağlık, Bilgi Teknolojisi, Olgunluk, Olgunluk Modeli, Olgunluk Faktörü, Temellendirilmiş Kuram, Sınıflandırma.

Maturity Factors of Health Information System and Technology

Samet Can[1], Ayça Tarhan[2]

[1,2]Bilgisayar Mühendisliği, Hacettepe Üniversitesi, 06800, Ankara, TÜRKİYE
[1]samet.can@hacettepe.edu.tr, [2]atarhan@hacettepe.edu.tr

Abstract. The quality and efficiency of service processes in the healthcare, which is dominated by dynamic, variable and multi-disciplinary processes, are directly affected by the maturity of information system and technology (IST) supporting these processes. Although there are many models proposed in the literature to evaluate or assure the maturity of health IST, there is no widely accepted model. In this paper, based on the 48 Health IST Maturity Models individually proposed in the mentioned studies, the factors for evaluate IST maturity in the field of health were extracted and classified by methodology of Grounded Theory. The classification and maturity factors proposed by this paper may be used as reference in their experimental studies for researchers and practitioners working on the maturity of health IST. In future studies, the factors extracted from this paper are

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

planned to be used as the basis for a survey to be conducted to understand the IST maturity of health organizations.

Keywords: Healthcare, Information Technology, Maturity, Maturity Model, Maturity Factor, Grounded Theory, Classification.

1 Giriş

Günümüzde bilgi teknolojisi, her alanda olduğu gibi sağlık alanında da önemli bir yer tutmaktadır. Bilgi Sistemi ve Teknolojisi (BST), bir organizasyonun etkinliğini ve verimliliğini artırmak amacıyla kullanılan önemli bir araçtır. Sağlık kurumlarında verilen hizmetlerin kalitesi, BST olgunluğuna göre farklılık göstermektedir. Hizmet kalitesini arttırmak amacıyla BST için yapılacak herhangi bir iyileştirme, hem hizmet alan bireyler hem de ilgili kurumlar için önemli faydalar sağlamaktadır.

"Olgunluk" kavramı ilk olarak Phillip Crosby [1] tarafından önerilmiş ve "tam, mükemmel veya hazır olma durumu" olarak tanımlanmıştır [2]. Daha genel bir bakış açısına göre bir olgunluk modeli, bir veya daha fazla iş alanındaki bir süreçler sınıfı için ayrı bir olgunluk seviyesi dizisinden oluşan ve bu süreçler için beklenen, istenen veya tipik bir evrimsel yolu temsil eden kavramsal bir çerçevedir [3]. Olgunluk modelleri, geçici veya olgunlaşmamış süreçleri, gelişmiş kalite ve yararlılık ile disiplinli ve olgun süreçler haline getirmek için bir gelişim yolu tanımlar. Bu sebeple olgunluk modelleri, organizasyonda var olan durumun tespiti ve yapılacak olan iyileştirmeleri belirleyebilmek açısından bizlere faydalar sağlamaktadır.

Sağlık hizmet süreçleri ve bilgi sistemleri/teknolojilerinin olgunluğunun değerlendirilmesi ve iyileştirilmesi amacıyla çeşitli olgunluk modelleri önerilmiştir [4]. Bu olgunluk modellerinin bazıları teletıp, dijital görüntüleme, resim arşivleme ve iletişim sistemleri (PACS) ve süreç yönetimi gibi belirli alt alanlara odaklanmaktadır. Bununla birlikte yaygın kabul görmüş bir model bulunmamaktadır. Bu alandaki geniş bilgi gözetildiğinde, mevcut modellerin ele aldığı faktörlerin çıkarılarak genel bir sınıflandırma yapılmasına ihtiyaç vardır. Bu ihtiyacı adreslemek üzere bu çalışmada, sağlık alanında önerilen veya kullanılan olgunluk modelleri üzerine derlenen kapsamlı bir çalışma havuzunu [5] referans seçtik. Bu havuzda yer alan kaynaklar arasından Bilgi Teknolojisi ile ilgili olanları alarak 48 kaynaklık bir alt havuz elde ettik. Elde ettiğimiz bu havuza Temellendirilmiş Kuram (İng. Grounded Theory) metodolojisini [6] uygulayarak faktör çıkarma ve sınıflandırma işlemini gerçekleştirdik.

Bu makalenin geri kalanı şu şekilde yapılandırılmıştır: Bölüm 2'de ilişkili çalışmalar özetlenmiştir. Bölüm 3'te Temellendirilmiş Kuram metodolojisiyle sağlık BST olgunluğuna yönelik çıkarılan faktörlerin sınıflandırması ve ilgili kategorilerin açıklamaları verilmiştir. Bölüm 4'te sonuç ve gelecek çalışmalara değinilmiştir.

2 İlişkili Çalışmalar

Literatür taramasında, bu çalışmadaki gibi sağlıkta olgunluk modelleri üzerine faktör çıkaran ve sınıflandırma yapan çalışmaları araştırdık. Araştırmalarımız sonucunda belirli alanlarda faktör çıkarımı yapan çalışmalar elde ettik. Bunlar daha çok kullanılabilirlik [8], olgunluk modeli [13] ve uzamsal teknoloji [14] gibi kısıtlı bir alan üzerine yoğunlaşan çalışmalardı. Tespit edilen ilişkili çalışmalar Tablo 1'de verilmiştir.

Bu çalışma, sağlıkta BST olgunluk modelleri üzerinden faktör çıkarımı yaptığı ve elde edilen faktörler ile bu alanda dar bir çerçevede kalmayıp daha geniş bir faktör setini adreslediği için diğer çalışmalardan farklılaşmaktadır.

Tablo 1. Sağlıkta olgunluk üzerine faktör çıkarımı yapan ilişkili çalışmalar

Referans	Yıl	Başlık	Araştırma Odağı
[7]	1996	An Examination of Information Systems and Technology Maturity and Its Relationship to Methods of Contributing Information to the Information Systems Planning Process in National Health Service Acute Trust Hospitals	Bilgi Sistemleri
[8]	2011	Promoting Usability in Health Organizations: Initial Steps and Progress Toward a Healthcare Usability Maturity Model	Kullanılabilirlik
[9]	2011	The Potential Role of Maturity Models in an Assessment of Hospital IT Capability in Ireland	Bilgi Teknolojisi
[10]	2013	Towards a Hospital Cooperation Maturity Model Construction and Evaluation of a Maturity Model and a Corresponding Tool for the Identification of Challenges and Success Factors for Cooperation in the Hospital Sector	İş Birliği
[11]	2014	Process Management in Hospitals: An Empirically Grounded Maturity Model	Süreç Yönetimi
[12]	2015	Health Information Technology (HIT) in Small and Medium Sized Physician Practices: Examination of Impacts and HIT Maturity	Bilgi Teknolojisi
[13]	2015	Quality System Maturity Model for Medical Devices-Medical Device Innovation Consortium (MDIC)	Olgunluk Modeli
[14]	2015	Spatial Maturity in a Health Agency: A Pilot Study	Uzamsal Teknoloji
[15]	2016	The Public Health Information Technology Maturity Index: An Approach to Evaluating the Adoption and Use of Public Health Information Technology	Bilgi Teknolojisi

3 Sağlık BST Olgunluk Faktörleri

3.1 Kaynak Havuzu

Sağlıkta olgunluk modelleri alanı, sağlık hizmetleri ve hastane yönetiminden, akademi veya uygulamanın iki tarafındaki bilgi teknolojisine kadar çeşitli paydaşları içerdiğinden tabiatı gereği disiplinler arasıdır. Ayrıca, bu alandaki bazı modeller gri literatürde

sunulmuş ve resmi olarak yayınlanmamıştır. Bu nedenle, bilimsel literatüre ek olarak gri literatürü de incelemeye dahil ederek doğal bir yaklaşım izleyen, sağlıkta olgunluk modelleri üzerine yürütülmüş bir çok-sesli literatür taramasının kaynak havuzunu [5] referans seçtik ve bu çalışmaya girdi olarak kullandık.

Referans olarak seçtiğimiz kaynak havuzunda yer alan çalışmaları, kapsanan teknolojik yönlere göre sınıflandırdık. Yaptığımız sınıflandırma sonucunda, Sağlık Bilgi Teknolojisi alanındaki 48 adet makaleyi faktör çıkarımı için temel aldık. Bu çalışma özelinde seçilen makaleleri içeren kaynak havuzuna ve çıkarılan bilgilerin detaylarına şu adreste yer alan çevrimiçi tablodan erişilebilir: https://bit.ly/2KrbIDl.

Tablo 2. Sağlık BST olgunluğu üzerine çalışma havuzu

#	**Makalenin Başlığı**	**Havuzdaki Referans #**
1	A focus area maturity model for a statewide master person index	1
2	A maturity model for BPM capability assessment in Dutch hospitals	3
3	A maturity model for care pathways	4
4	A maturity model for telemedicine implementation	5
5	A PACS maturity model A systematic meta-analytic review on maturation and evolvability of PACS in the hospital enterprise	7
6	A patient-centered framework for evaluating digital maturity of health services: A systematic review	8
7	Act on oncology' as a new comprehensive approach to assess prostate cancer centres - Method description and results of a pilot study	10
8	An Examination of Information Systems and Technology Maturity and it's relationship to Methods of Contributing Information to the Information Systems Planning Process in National Health Service Acute Trust Hospitals	11
9	An Open Government Maturity Model for social media-based public engagement	13
10	Audit of health agency for supporting an e-health management system (EHMS) case study: Ciparay Health Center, Bandung	18
11	Capabilities and levels of maturity in IT-based case management	19
12	Cloud maturity model for the healthcare industry	21
13	Developing a SSE-CMM-based security risk assessment process for patient-centered healthcare systems	25
14	Development of a maturity model for telemedicine	27
15	E-healthcare maturity in Taiwan	30
16	Electronic Healthcare Maturity Model (eHMM)	32
17	Enhancing benefits from healthcare IT adoption using design science research: Presenting a unified application of the IT Capability Maturity Framework and the Electronic Medical Record Adoption Model	34
18	Enterprise Architecture Maturity: The Story of the Veterans Health Administration	35
19	Evolutionistic or revolutionary paths: A PACS maturity model for strategic situational planning	38
20	Game maturity model for health care	39
21	HCMM - A maturity model for measuring and assessing the quality of cooperation between and within hospitals	41
22	Health information technology (HIT) in small and medium sized physician practices: Examination of impacts and HIT maturity	43
23	Health: IT leader or laggard? A comparative assessment of IT maturity	44
24	Healthcare breach security Maturity Model	46
25	Healthcare Information Security Adoption Model (HISAM)	47
26	Healthcare quality maturity assessment model based on quality drivers	48
27	High-reliability health care: Getting there from here	49

28	Hospital medicine maturity framework: proposing a novel framework for defining stages of hospital medicine in Canada	50
29	Identity Management for Health Professionals	51
30	Informatics Capability Maturity Model (ICMM)	52
31	Information management and technology strategy development in the UK's acute hospital sector: a maturity model perspective	53
32	Information technology capacities assessment tool in hospitals: Instrument development and validation	54
33	Maturity Model for Integrated Care	57
34	Measuring Maturity of Use for Electronic Medical Records (EMRs) in British Columbia: The Physician Information Technology Office (PITO)	59
35	Medicaid Information Technology Architecture (MITA) Maturity Model	62
36	Process management in hospitals: an empirically grounded maturity model	67
37	Procurement maturity, alignment and performance: a Dutch hospital case comparison	68
38	Promoting usability in health organizations: Initial steps and progress toward a healthcare usability maturity model	69
39	Public Private Partnership in Italian Health Care Management: An Organizational Maturity Assessment Model	70
40	Quality System Maturity Model for Medical Devices-Medical Device Innovation Consortium (MDIC)	72
41	Spatial Maturity in a Health Agency: A Pilot Study	75
42	The Healthcare Paperless Maturity Model	77
43	The Healthcare Security Maturity Model	78
44	The potential role of Maturity Models in an assessment of Hospital I.T. Capability in Ireland	79
45	The Public Health Information Technology Maturity Index: An Approach to Evaluating the Adoption and Use of Public Health Information Technology	80
46	The State of Nursing Home Information Technology Sophistication in Rural and Nonrural US Markets	82
47	The Telemedicine Service Maturity Model- A Framework for the Measurement and Improvement of Telemedicine Services	83
48	Towards a Hospital Cooperation Maturity Model - Construction and Evaluation of a Maturity Model and a corresponding Tool for the Identification of Challenges and Success Factors for Cooperation in the Hospital Sector	84

3.2 Olgunluk Faktörleri

İkinci Bölümde bahsedildiği gibi, sağlık alanında olgunluğa yönelik olarak faktör çıkaran bazı çalışmalar mevcuttur. Ne var ki bunlar alana yönelik bireysel katkılarına odaklanmakta, mevcut eğilimleri ve gelecekteki muhtemel yönelimleri sınıflandırma konusunda genel bir bakış vermemektedir. Bu boşluğu adreslemek üzere bu çalışmada, bir alan hakkında genel bir anlayış oluşturmak için en uygun yöntem olarak gösterilen Temellendirilmiş Kuram [16] seçilmiştir. Bu yöntem, "temel bir alan hakkında tümevarımsal bir teori üretmek için, sistematik olarak uygulanan bir dizi aşamayı kullanan, veri toplama ile bağlantılı genel bir analiz metodolojisi" olarak tanımlanır [17]. Temellendirilmiş Kuram, üç ana aşamadan oluşur: (1) kodlama, (2) not alma ve teori oluşturma ve (3) teoriyi bütünleştirme, arındırma ve yazma [17]. Takip etmeyi seçtiğimiz Glaser'in yöntemine [18] göre Temellendirilmiş Kuramda, bir araştırma problemi tanımlamaksızın araştırmaya başlanması tavsiye edilir. Kavramsal bir model geliştirmenin amacı, Temellendirilmiş Kuram için uygun olan esnek bir şekilde ortaya çıkacak model ve kategorileri tespit etmektir [16].

Temellendirilmiş Kuramda kullanılan üç terim vardır: kod, kavram ve kategori [18]. Kod ifadelerden alınan temel gözlemdir, kavram kod grubudur ve kategori ise kavram grubudur [16]. Metodolojinin ilk aşaması olan kodlamada, ilk olarak araştırmaya odaklanılacak temel kategori ve kavramları tanımlamak için açık bir kodlama yapılır ve ikinci olarak temel kategoriyi geliştirmek için seçici bir kodlama yapılır. Daha sonra not alma aşamasında, her bir kavram için belirlenmiş kavramlar ve notlar, ana kategoriyi şekillendirmek için birbirleriyle karşılaştırılır. Son aşamada tüm bulgular karşılaştırmalı bir yöntemle birleştirilerek birbirine bağlanır ve bir teori oluşturulur [17]. Bu çalışmada yalnızca süreçten çıkan temel kategorilerle ilgilendiğimiz için, bu son aşama uygulanmamıştır.

Temellendirilmiş Kuram, çoğunlukla anket, görüşme veya gözlem formundaki verilerle kullanılsa da kodlar ifadelerden satır satır çıkarılmaktadır. Kodlamalar, makaleler arasındaki olası bir terminoloji farkını ortadan kaldırmak için karşılaştırmalı yöntemler izlenerek gerçekleştirilir, yani ilgili makaleler birlikte gruplanır ve yazarlar tarafından farklı şekilde kullanılmış olabilecek kesin terminolojileri tanımlamak ve çıkarmak için tekrar kodlanır [16]. Bu çalışmada, kaynak havuzuna seçtiğimiz 48 adet makaleyi girdi olarak kullandık. Başlangıçta yerleşmiş kodlarımız olmadığı ve ilgili girdiyi kaybetme riskini göze alamadığımız için her makaleyi en az bir kez değerlendirdik ve olgunluk modelleri ile ele alınan unsurlara ve boyutlara kodlar atadık. Sonra kodlardan gelen kavramları daha yüksek bir gruplama unsuru olarak belirledik ve son olarak kategorilere karar verdik. Bu şekilde Temellendirilmiş Kuram metodolojisine göre, olgunluk modellerinde yer alan 128 tane olgunluk faktörü çıkarttık ve bu faktörleri gruplayarak ana kategorileri belirledik. Belirlenen kategoriler altında alt-faktörlere de yer verdik. Tablo 3'te, iki ve üzeri kaynaktan referans alınan faktörler, frekansına göre büyükten küçüğe doğru verilmiştir. Ana kategorilerin alt-faktörlere kırınımı ve frekansları ise Ek-1'de sunulmuştur.

Tablo 3. Frekansa göre sıralanmış faktörler ve referans çalışmalar

Faktör	**Referans Çalışmalar**	**Frekans**
Information Technology	3, 5, 27, 51, 52, 67, 68, 79	8
Culture	3, 4, 44, 48, 67, 68	6
Finance	1, 18, 27, 57	4
Policy and Legislation	5, 27, 68, 80	4
Strategy	44, 67, 68, 75	4
Collaboration	19, 43, 75	3
People	3, 48, 68	3
Quality	41, 48, 51	3
Governance	1, 3, 57	3
Integration	30, 43, 79	3
Organizational Processes	5, 68, 69	3
Standardization	41, 57	3
Resources	69, 75, 80	3
Communication	43, 44	2
Involvement	41, 43	2
Users	27, 83	2
Completeness	41, 84	2
Ease of Integration	41, 84	2
Financial Sustainability	5, 83	2
Funding	57, 75	2

Liability	41, 84	2
Responsibilities	41, 51	2
Information	19, 57	2
Documentation	41, 84	2
Availability	41, 84	2
Administration	19, 79	2
Infrastructure	4, 69	2
Technical Infrastructure	1, 44	2
Innovation	21, 80	2
Change Management	52, 83	2
Structure	57, 67, 75	2

Aşağıda 48 adet sağlık BST olgunluk modelinden çıkardığımız faktörlerden grupladığımız kategoriler ve açıklamaları yer almaktadır.

Kültür (Culture): Bu faktör, bir hastanenin etkili süreç yönetimi kurması için gerekli olan iletişim ve liderlikle ilgili uygulamaları kapsar. Bu uygulamalar, işlevsel sınırlar ve hiyerarşik seviyeler arasında açık iletişimi ve ayrıca klinik ve idari personelin yetkilendirilmesini içerir.

Strateji (Strategy): Bu faktör, stratejik hedeflerin tanımlanmasında süreç yönetiminin gelişimi için ön koşul olan ilkeleri kapsar. Bunlar arasında bölümler arası iş birliği ve bilgi alışverişi de bulunmaktadır.

Yapı (Structure): Bu faktör, bölümler (klinikler) arasındaki küçük bariyerlerin kaldırılması ve uzmanlıkların ekipler arasında düzgün dağıtılması gibi süreç yönetimi için gerekli olan organizasyonel boyutları içerir.

Süreç (Process): Bu faktör, süreç performans incelemesi, süreç dokümantasyonu ve hasta akış oryantasyonu gibi, hastanelerde süreç yönetimi için önemli olan çalışma pratiklerini içerir.

Bilgi (Information): Bu faktör, hastanede kaliteli bilgilerin toplanması, yönetimi ve paylaşılmasıyla ilgilidir. Bu alandaki yetenek, kullanıcıların doğru bilgiye doğru zamanda erişebilmeleri anlamına gelir.

Bilgi Teknolojisi (Information Technology): Bu faktör, hastanede kullanılan BST'nin, hasta bakım akışını ne kadar düzgün bir şekilde destekleyebildiği, belirli bir kullanım bağlamında etkinlik, verimlilik ve memnuniyet kriterlerine göre belirlenen hedeflere ulaşmak için kullanıcılar tarafından ne ölçüde kullanılabileceği ve verilerin yeterli bir şekilde bulunup bulunmadığını araştırmaya yönelik maddeler içerir.

Yönetişim (Governance): Bölgesel veya ulusal düzeyde sağlık hizmetlerini sağlamak için gerekli olan geniş kapsamlı değişiklikler önemli bir zorluktur. Etkili değişim yönetimi, finansman, iletişim ve yeni çalışma pratiklerini uygulama ve zorunlu kılma gücü ile birlikte çok yıllık programlara ihtiyaç duyar. Yönetişim, farklı organizasyonlar ve meslekler arasında amacın uyumu ve iş birliği yapma ve genel sağlık sisteminin çıkarlarını bireysel teşviklerin üstünde tutmaya istekli olma anlamına gelir. Teknolojiye dayalı sağlık hizmetlerini, sağlık profesyonelleri ve hastalar için kullanımı kolay, güvenilir ve kabul edilebilir kılacak şekilde yönetmek anlamına da gelir.

Standardizasyon (Standardization): Sağlık sistemlerinde bilgi paylaşımı ve iş birliği ihtiyaçları göz önüne alındığında, kullanılan farklı sistemlerin sayısı ve verilerin depolandığı biçimlerin basitleştirilmesi durumunda işler daha kolay hale getirilebilir. Standardizasyon, veri merkezlerinin birleştirilmesi, daha az sistemde standardizasyon yapılması ve bir bölgede hangi teknik standartların kullanılacağına karar verilmesi anlamına gelir.

Finans (Finance): Daha iyi bir entegrasyon sağlayabilmek için sağlık sistemlerini değiştirmek, başlangıç yatırımı ve finansman gerektirir. Başlangıç yatırımı ve devam eden maliyetlerin finanse edilebildiğinden emin olmak, bölgesel/ulusal bütçelerden teşvik edici fonlara, Avrupa Birliği yatırım fonlarından kamu-özel ortaklıkları ve risk paylaşım mekanizmalarına kadar çeşitli mekanizmaları kullanan önemli bir faaliyettir.

Değerlendirme Yöntemleri (Evaluation Methods): Sağlık sistemlerini desteklemek için yeni servisler getirildikçe sağlık hizmetlerinin kalitesi, maliyeti, erişimi ve hasta deneyimi üzerinde istenen etkinin gerçekleşip gerçekleşmediğini sağlamaya ihtiyaç vardır. Değerlendirme yöntemleri, her değişikliğin etkisinin değerlendirildiği kanıta dayalı yatırım kavramını destekler. Burada Sağlık BST Değerlendirmesi önemli bir yöntemdir ve sağlık sistemlerinin bölgesel veya ulusal seviyeye yükseltilmesinin maliyetini haklı çıkarmak için kullanılabilir.

İnovasyon (Innovation): En iyi fikirlerin çoğu, mevcut süreçlerin neresinde iyileştirmeler yapılabileceğini anlayan klinisyenler, hemşireler ve sosyal hizmet uzmanlarından gelmesi muhtemeldir. Bu yeniliklerin tanınması, değerlendirilmesi ve mümkünse sistem genelinde fayda sağlayacak şekilde arttırılması gerekir. İnovasyon yönetimi, sağlık BST için en iyi sonuçları almak amacıyla inovasyon sürecini yönetmek ve iyi fikirlerin teşvik edilmesini ve ödüllendirilmesini sağlamak anlamına gelir.

Kapasite Geliştirme (Capacity Building): Kapasite geliştirme, bireylerin ve kuruluşların işlerini ustalıkla yapabilmeleri için gereken beceri ve bilgileri edinmeleri, iyileştirmeleri ve elde tutmalarıdır. Sağlık sistemlerinin kaliteyi, maliyeti ve erişimi geliştirmek için sürekli olarak çabalayan öğrenme sistemleri olması gerekir. Talepler değişmeye devam ettikçe, beceriler, yetenekler ve deneyimler korunmalıdır. Kapasite geliştirme, bir sonraki proje setini geliştirmek için bilginin alınmasını ve kullanılmasını sağlamak, daha fazla üretkenlik ve artan başarıya ulaşmak anlamına gelir.

4 Sonuçlar ve Gelecek Çalışmalar

Sağlık hizmetleri alanının karmaşık ve çok disiplinli yapısı, süreç ve insan yönetimi ile bilgi sistemi ve teknolojisi açısından önemli zorluklar doğurmaktadır. Dahası, sağlık hizmetlerinin dijitalleştirilme miktarını arttırmak amacıyla geçiş sürecini rasyonelleştirmek ve etkin bir şekilde yönetmek için iyi tanımlanmış kurallara ihtiyaç vardır. Bu zorluklara cevap olarak, sağlık alanında çok çeşitli olgunluk modelleri önerilmiş ve uygulanmıştır.

Bu çalışmada, çok-sesli literatür tarama uygulanarak sağlıkta olgunluk modelleri üzerine oluşturulmuş bir çalışma havuzundan, sağlık BST olgunluğunu adresleyen 48

makale seçilmiştir. Ardından, bu alanda dikkate alınması gereken faktör kategorileri üzerine genel bir fikir oluşturmak amacıyla, Temellendirilmiş Kuram metodolojisiyle sağlık BST olgunluğu üzerine faktör çıkarma ve sınıflandırma yapılarak elde edilen temel kategoriler tanımlanmıştır.

Bu çalışma ile önerilen sınıflandırma, sağlık BST olgunluğu üzerine yapılacak yatırım, uygulama ve araştırmalar için yöneticilere, pratisyenlere ve araştırmacılara referans olabilir. Gelecek çalışmalarda, bu çalışma ile çıkarılan faktörlerin ve sınıflarının, sağlık kurumlarının BST olgunluğunu anlamak amacıyla yapılacak bir ankete temel alınması planlanmaktadır.

Ek-1

Tablo 4. Sağlık BST Olgunluk Faktörleri ve Alt-Faktörleri

Faktör	*Referans Çalışmalar*	*Frekans*
Culture	**3, 4, 44, 48, 67, 68**	**6**
Awareness	75	1
User Awareness	30	1
Citizen Empowerment	57	1
Collaboration	19, 43, 75	3
Communication	43, 44	2
Interaction/Involvement with Community	5	1
Involvement	41, 43	2
Involvement of Employees	84	1
People	3, 48, 68	3
Customer	18	1
IS Staff	30	1
Users	27, 83	2
Focus on Users	69	1
Individual Users	5	1
Safety Culture	49	1
Strategy	**44, 67, 68, 75**	**4**
Internal	18	1
Population Approach	57	1
Strategic Alignment	3	1
Aligning Business & Informatics	52	1
Structure	**57, 67, 75**	**2**
Decisions	19	1
Business Intelligence	52	1
Organization	51	1
Resources	69, 75, 80	3
Process	**69**	**1**
Accuracy of Process Results	62	1
Cost Effectiveness	62	1
Design	4	1
Efficiency	62	1
IS Planning and Control	30	1
Methods	3	1

Operation	48	1
Organizational Processes	5, 68, 69	3
Performance Management	4	1
Monitoring and Control	68	1
Robust Process Improvement	49	1
Timeliness of Process	62	1
Utility or Value to Stakeholders	62	1
Workflows	75	1
Optimization	21	1
Information	**19, 57**	**2**
Access	75	1
Data Access and Accuracy	62	1
Documentation	41, 84	2
Fit for Purpose	75	1
General Information and Statistics	44	1
Information Management	52	1
Information Resource	44	1
Case History	19	1
Patient Consent	1	1
Relevance	75	1
Information Technology	**3, 5, 27, 51, 52, 67, 68, 79**	**8**
Application Portfolio	30	1
Availability	41, 84	2
Capacity	75	1
Functional IT Capacities	54	1
Technological IT Capacities	54	1
EHR Systems	83	1
Administration	19, 79	2
Clinical	79	1
Clinical Support	79	1
eHealth Services	57	1
Patient Management	79	1
Infrastructure	4, 69	2
Community Infrastructure	80	1
Foundation Building	21	1
ICT Infrastructure	83	1
IT Infrastructure	30	1
Technical Infrastructure	1, 44	2
Integration	30, 43, 79	3
Degree of Integration	82	1
External Integration	54	1
Internal Integration	54	1
IT Capabilities	82	1
Governance	**1, 3, 57**	**3**
Leadership	49	1
Top Managers	48	1
Liability	41, 84	2
Management	69	1
Owner & Performers	4	1
Removal of Inhibitors	57	1

Responsibilities	41, 51	2
Clear Responsibilities	84	1
Standardization	**41, 57**	**3**
Degree of Standardization	84	1
Interoperability and Standards	44, 80	1
Legal Authority	1	1
Policy and Legislation	5, 27, 68, 80	4
Policy Implications	49	1
Privacy and Security	80	1
Procedures	27	1
Finance	**1, 18, 27, 57**	**4**
Financial Sustainability	5, 83	2
Funding	57, 75	2
Evaluation Methods	**57**	**1**
Completeness	41, 84	2
Ease of Integration	41, 84	2
Extent of IT Use	82	1
Quality	41, 48, 51	3
Quality Focus	48	1
Redundancy	41	1
Negative Redundancy	84	1
Scale and Scope of Use	80	1
Stability	43	1
Support	41	1
Willingness to Support	84	1
System Use	43	1
Technology Use	50	1
User Satisfaction	43	1
Vendor Effectiveness	44	1
Innovation	**21, 80**	**2**
Breadth of Ambition	57	1
Change Management	52, 83	2
Readiness to Change	57	1
Innovation Management	57	1
Capacity Building	**57**	**1**
Accreditation	48	1
Competency	80	1
Future Research and Development Tasks	49	1
Learning and Growth	18	1
Education	69	1
Experimentation	21	1
Personnel Capabilities	41	1
Qualifications & Skills of Employees	84	1
Practices	67	1
Training	80	1

Kaynakça

1. Crosby, P.: Quality is Free, McGraw-Hill (1979).
2. Stevenson, A. & Waite, M.: Oxford English Dictionary – The Definitive Record of the English Language. Oxford University Publisher (2004).
3. Becker, J., Knackstedt, R. & Poeppelbuss, J.: Developing Maturity Models for IT Management - A Procedure Model and its Application. 1, 213-222 (2009).
4. Söylemez, M., Tarhan, A.: The Use of Maturity/Capability Frameworks for Healthcare Process Assessment and Improvement. in proceedings of the 16th International SPICE Conference, Springer CCIS (Communications in Computer and Information Science), vol. 609, pp. 31-42 (2016).
5. Tarhan, A., Garousi, V., Turetken, O., Soylemez, M., Garossi, S.: Dataset for paper: Maturity assessment and maturity models in healthcare: a multivocal literature review. http://doi.org/10.5281/zenodo.2600722 (2019).
6. Glaser, B., Strauss, A. L.: The Discovery of Grounded Theory Strategies for Qualitative Research. Aldine Publishing Company, Chicago (1967).
7. Mitchell, I. J.: An examination of information systems and technology maturity and its relationship to methods of contributing information to the information systems planning process in National Health Service acute trust hospitals (1996).
8. Force, H. U. T.: Promoting Usability in Health Organizations: initial steps and progress toward a healthcare usability maturity model. Healthcare Information and Management Systems Society, http://s3.amazonaws.com/rdcms-himss/files/production/public/HIMSSorg/Content/files/HIMSS_Promoting_Usability_in_Health_Org.pdf (2011).
9. O'Neill, D.: The potential role of Maturity Models in an assessment of Hospital IT Capability in Ireland, Citeseer (2013).
10. Blondiau, A., Cleven, A., Rohner, P.: Towards a Hospital Cooperation Maturity Model-Construction and Evaluation of a Maturity Model and a corresponding Tool for the Identification of Challenges and Success Factors for Cooperation in the Hospital Sector. Int. Journal Health Information Management Research, vol. 1, no. 1, pp. 4–14 (2013).
11. Cleven, A. K., Winter, R., Wortmann, F., Mettler, T.: Process management in hospitals: an empirically grounded maturity model. Business Research, vol. 7, no. 2, pp. 191–216 (2014).
12. Wallace, S. A.: Health information technology (HIT) in small and medium sized physician practices: examination of impacts and HIT maturity (2015).
13. MDIC: Quality System Maturity Model for Medical Devices. Medical Device Innovation Consortium (MDIC), http://mdic.org/wp-content/uploads/2015/06/MDIC-Deloitte_Maturity-Model-Research-Report_June-2015.pdf (2015).
14. Gudes, O., Mullan, N., Weeramanthri, T.: Spatial Maturity in a Health Agency : A Pilot Study. https://www.crcsi.com.au/assets/Resources/CRCSI-Spatial-Maturity-in-a-Health-Agency-Report-June2015.pdf (2015).
15. Crowley, K., Gold, R. S., Bandi, S., Agarwal, R.: The Public Health Information Technology Maturity Index: An Approach to Evaluating the Adoption and Use of Public Health Information Technology. Front. Public Heal. Serv. Syst. Res., vol. 5, no. 2, pp. 26–33, (2016).
16. Stray, V., Sjøberg, D. I. K, Dybå, T.: The daily stand-up meeting: A grounded theory study. J. Syst. Softw., vol. 114, pp. 101–124 (2016).
17. Glaser, B.: Basics of grounded theory analysis: Emergence vs forcing (1992).
18. Glaser, B.: Getting out of the data: Grounded theory conceptualization (2011).

Mikroservis Mimari ile Geliştirilmiş bir Uygulamanın Mevcut Çoklu Uygulama Portföyüne Entegre Edilmesi

Ümit Kanoğlu[1[0000-0001-8245-1355]] , Ali İmre[1[0000-0001-9477-0660]] ve Oumout Chouseinoglou[2[0000-0002-8513-351X]]

[1] Türk Telekom, Ankara, Türkiye
{umit.kanoglu, ali.imre}@turktelekom.com.tr
[2] Endüstri Mühendisliği, Hacettepe Üniversitesi, Ankara, Türkiye
uhus@hacettepe.edu.tr

Özet. Servis kalitesinin her zaman en üst seviyede olması gereken kurumlarda bilişim sistemleri yeterli erişilebilirlik olgunluğunu 7/24 sergilemeli, erişilebilirlik seviyesinin yanında fonksiyonel ve regülatif anlamda da sürekli güncel kalmaları gerekmektedir. Her ne kadar bu güncellemeler sürüm planlama süreçleri işletilerek yönetilse de, genellikle monolitik mimari üzerine inşa edilmiş mevcut uygulamaların devreye alımı sonrasında canlı sistemlerde servis kesintileri görülebilmektedir. Bu çalışmada Türk Telekom'un kesintisiz devreye alımları sağlamak için iç müşterinin kullanımına sunduğu ve ilk defa mikroservis mimarinin yanı sıra davranış güdümlü geliştirme yönteminin de kullanıldığı bir projenin kurum envanterine ve mevcut süreçlere entegrasyonu özetlenmiştir.

Anahtar Kelimeler: Mikroservis, Davranış Güdümlü Geliştirme, Konteyner, Konteyner Orkestrasyonu, Sürekli Entegrasyon, Sürekli Devreye Alım.

Integration of an Application Developed on Microservice Architecture to the Existing Multiapplication Portfolio

Ümit Kanoğlu[1[0000-0001-8245-1355]] , Ali İmre[1[0000-0001-9477-0660]] and Oumout Chouseinoglou[2[0000-0002-8513-351X]]

[1] Türk Telekom, Ankara, Turkey
{umit.kanoglu, ali.imre}@turktelekom.com.tr
[2] Department of Industrial Engineering, Hacettepe University, Ankara, Turkey
uhus@hacettepe.edu.tr

Abstract. Companies, where the service level is required to be at the highest level at all times, must have information systems that not only display an adequate level of access maturity continuously but also are up-to-date both in functional and regulative terms. Even though these updates are managed by

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

version planning processes, service loses on live systems may occur, especially in legacy applications built on top of monolithic architectures. In this study, a project presented by Türk Telekom to its internal customers, enabling continuous deployment by using for the first time a microservice architecture and behavior driven development, is summarized together with the details of integrating it into the corporate application inventory and existing corporate processes.

Keywords: Microservice, Behavior Driven Development, Container, Container Orchestration, Continuous Integration, Continuous Deployment.

1 Giriş

Türk Telekom'un [1] abonelerine ve iç müşterilerine kesintisiz hizmet sağlamak amacıyla oluşturduğu yazılım uygulama portföyünün tamamına yakını monolitik mimari ile geliştirilmiş uygulamalardan oluşmaktadır. Monolitik mimari ile geliştirilmiş ve iş süreçleri içerisinde birbirlerine bağımlılıkları yüksek olan kurum uygulamalarının güncellenmesinde, yazılım geliştirme ve operasyonel süreçlerde farklı türde işletim zorlukları ile karşılaşılmaktadır. Bunlar; en küçük kod değişikliğinde tüm kod bloğunun tekrar derlenmesi, devreye alım işlemi esnasında canlı ortam üzerinde çalışan ilgili uygulama sunucularının kapatılması ve uygulamaların devreye alınmasının zaman ve erişilebilirlik anlamında hizmet kayıplarına neden olmasıdır.

Son yıllarda ticari rekabetin artmasıyla kesintisiz ve sık devreye alım yetenekleri önem kazanmış, monolitik mimariye alternatif olarak sürekli entegrasyon (Continuous Integration-CI), sürekli devreye alım (Continuous Deployment-CD) ve mikroservis mimari (Microservices Architecture-MSA) gibi yaklaşımların üzerinde daha fazla durulması gerekliliği ortaya çıkmıştır [2]. Mikroservis, genellikle tek bir işlemi yapan birbirinden bağımsız olarak kendi sürecinde çalışan iş (kod) paketleri üzerine kurgulanmış mimari bir yaklaşımdır [3]. Mikroservisler iş yetenekleri doğrultusunda birbirilerinden etkilenmeden devreye alımı sağlayacak bir şekilde tasarlanmaktadır [3,4] ve birbirleri arasında senkron veya asenkron iletişim kurabilirler. Mikroservisler uygulama sunucularında konteynerler üzerinde konumlandırılır ve genellikle dağıtık olarak yönetilir. Konteyner, bir uygulamanın kodunun yanı sıra bağımlılık ve yapılandırmalarını tutarlı ve verimli bir şekilde paketleyen sanallaştırma tekniği olup, yazılımı çalıştırmak için gereken kütüphaneler ve konfigürasyonları paketlemeye imkân tanır [5]. Konteynerler, MSA üzerinde yazılım geliştirme, sunma ve sürdürme için kullanılmaktadır. Konteyner yönetimi, çoklu kümeler (cluster) içindeki mikroservisler üzerine kurulmuş ayrı ayrı konteynerlerin yönetilmesi veya programlanması için otomatik bir işlemdir [4]. Bu çalışma kapsamında incelenen projede kullanılan bir diğer yazılım geliştirme yaklaşımı da davranış güdümlü geliştirme (Behavior Driven Development-BDD) olup, temel amacı müşterinin talep ettiği ürünün ya da fonksiyonun kullanım senaryosu olarak belirlediği davranışları üzerinden geliştirme aşamalarında ilerleme sağlamaktır [6]. Bu yöntemde müşteri kabul kriterleri ön planda olup kabul kriterlerinden kaynak koda giden süreç için belli araçlar kullanılarak test merkezli geliştirme yapılmaktadır. Bu kavramların yanında CI, kod üzerinde yapılan her değişikliğin ardından, tüm sistemin çalışır durumda olduğunu,

yapılan değişikliğin sistemin bazı bölümlerinde kırılmalara yol açmadığını tespit etmek için kullanılan yöntem olup CD ise, başarılı olan bir kod derleme işleminin uygulama sunucusu ya da sunucularına otomatik olarak yaygınlaştırılmasıdır [7].

Service Level Agreement (SLA) projesi, Türk Telekom'da MSA, CI ve CD yaklaşımlarının kurum içi uygulanabilirliğine yönelik bilincin arttırılması, aynı zamanda bahsi geçen yaklaşımların kurum içinde uygulanma aşamalarında karşılaşılan zorlukların ve öğrenilmiş derslerin değerlendirilmesi açısından pilot bir proje olarak devreye alınmıştır. Bu deneyim bildirisinin geri kalan kısmında, Bölüm 2'de proje deneyimi, Bölüm 3'te öğrenilmiş dersler ve Bölüm 4'te çalışmadan tecrübe edilmiş sonuçlar verilmektedir.

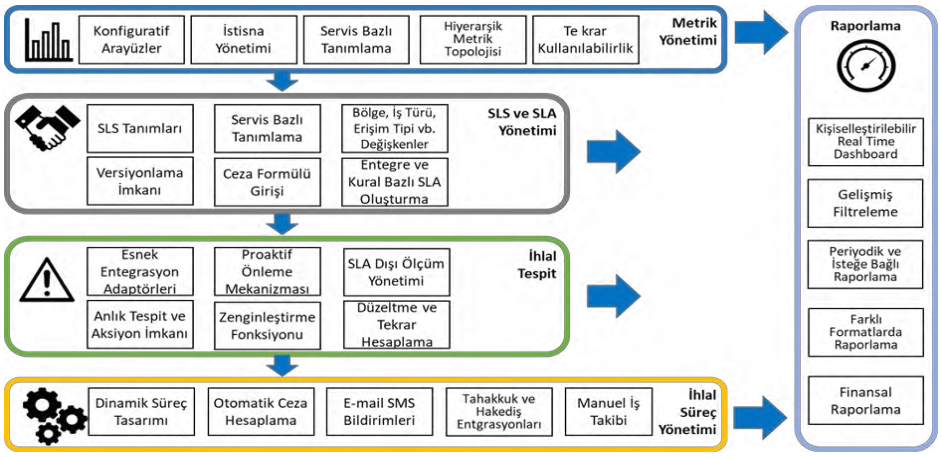
2 Proje Deneyimi

SLA yönetimi uygulamasının MSA'ye uygun geliştirilmesi bu çalışmada bir vaka olarak ele alınmış olup, proje sürecinde elde edilmiş olan öğrenme çıktıları bu çalışmanın odak noktasını oluşturmaktadır. SLA uygulamasının kapsamı ses, Internet ve veri ürünleri ile ilgili tek bir uygulama üzerinden satış ve müşteri şikâyet süreçlerinin SLA takibinin yapılmasıdır. Müşteriye bir ürün satıldığı zaman, müşteriye verilen taahhütler hem ürünün tesis ve şikâyet süreci ile hem de ürünün kullanılabilirliği ile ilgilidir. Uygulamanın geliştirilmesi ve bir pilot hizmet için devreye alım süreci 01.11.2017 ile 01.10.2018 arasında gerçekleşmiştir. İlk fazda uygulamanın temel fonksiyonlarının geliştirilmesi planlanmış, uygulama kuruma entegre edildikten sonra satılan tüm hizmetler fazlara ayrılarak uygulama kapsamına alınması hedeflenmiştir. Proje ilk fazı Java ile MongoDB ve Oracle veritabanları üzerinde geliştirilmiştir, geliştirme eforu 2.200 kişi/gün olup yaklaşık 600.000 satır koddan oluşmaktadır. Kurum bünyesinde, telekom sektörü standartlarını belirten TMForum'a [8] göre oluşturulmuş bir uygulama portföyü bulunmakta ve uygulamalar müşteri, sipariş, problem, envanter gibi alanlara ayrıştırılarak geliştirilmektedir. Yeni bir uygulamanın mevcut uygulamalarla entegre şekilde çalışması gerektiğinden, mevcut uygulamalarla entegrasyon etkisi oluşmakta ve geliştirme yapılması gereken uygulama sayısı artmaktadır.

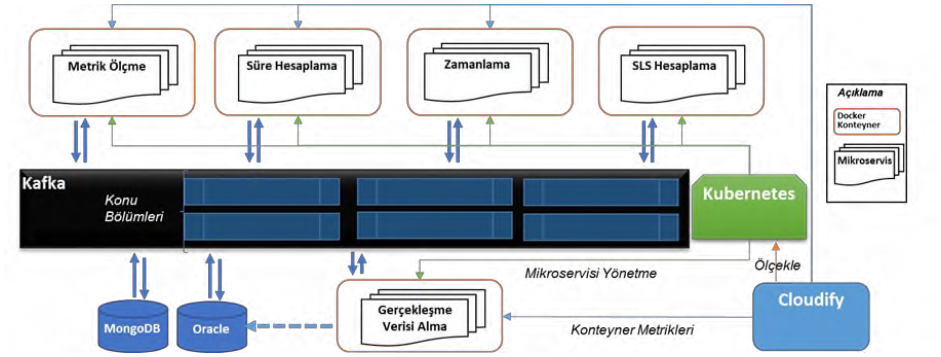
2.1 Uygulanan Mikroservis Mimari (MSA)

Uygulama, MSA'e uygun olarak geliştirilmiş olup, Şekil 1'de gösterildiği gibi entegrasyon, ölçme, hesaplama, doğrulama, raporlama ve grafik görseller gibi SLA yönetiminin alt süreçleri mantıksal olarak ayrıştırılarak mikroservisler oluşturulmuştur. Burada oklar takip edilmekte olan genel iş süreci akışını gösternmektedir. Kurumda yazılım geliştirme süreçlerinde tecrübe edilen temel problemlerden biri uygulamanın mevcut uygulamalarla entegre edilmesidir. MSA ile uygulamanın entegrasyon noktası ayrı bir mikroservis olarak yazılarak geliştirme sürecinde entegrasyonda yaşanan zorluk ve aksamalar uygulamanın temel fonksiyonlarından soyut yönetilebilmiştir. MSA'da, mikroservisler kod düzeyinden veritabanı seviyesine kadar birbirinden soyut şekilde oluşturulmalıdır. Fakat pratikte bunu kısıtlayan etkenler olabilir. Şekil 2'de verildiği üzere, geliştirilen uygulama için tüm mikroservisler ayrı ayrı kodlanmış ama

veri bütünlüğü önemli olduğu için veritabanı seviyesinde fonksiyonel olarak mikroservis ayrımına gidilmemiştir. Mikroservislerin kendi içerisindeki entegrasyonu Kafka[1] ile sağlanmıştır. Fonksiyonel açıdan tek veritabanı olup, uygulama tanım ve parametreleri için ayrı veritabanı oluşturulmuştur, MSA ile CI ve CD süreçleri desteklenmiştir. Her bir mikroservis için ayrı yönetilen geliştirme ve teslimatlar CI ile yönetilerek, DevOps yaklaşıma uygun bir süreç işletilmiştir. Mikroservisler, Docker[2] konteyner yapıdaki sunucular üzerinde Kubernetes[3] ile orkestra edilmiştir.



Şekil 1. MSA Tasarımı



Şekil 2. SLA Uygulamasının MSA'i

[1] https://kafka.apache.org/
[2] https://www.docker.com/
[3] https://kubernetes.io/

2.2 Uygulanan Davranış Güdümlü Geliştirme (BDD), Sürekli Entegrasyon (CI) ve Sürekli Devreye Alma (CD)

Uygulama 22 mikroservisten oluşturulmuştur. Tablo 1'de verilen bu mikroservislerden 12 tanesi için BDD aracı ile yaklaşık 500 adet test senaryosunun otomatik koşulması bir saat sürmekte, zamanlanarak otomatik koşulabilmekte ve sonuçları e-posta ile iletilmektedir. BDD aracı olarak Cucumber[4] kullanılmış, Web servis ve fonksiyonel mikroservislerin davranışlarını %100 kapsayacak şekilde otomatik testler oluşturulmuştur. Dolayısıyla, mikroservislerde değişiklik olduğunda CI ile otomatik olarak canlı ortamda güncellenebilmektedir. "*Happy path*" olarak kapsanan mikroservisler hata olma olasılığı düşük olanlardır. Otomatik test kapsamına alınmayan 10 mikroservis ise kullanıcı arayüzü ve raporlamalar ile ilgili olanlardır.

Mikroservis yaklaşımda, CI ve CD araçları ile uçtan uca, kaynak kod seviyesinden canlı ortama yüklemeye kadar tüm süreçlerin otomasyon dâhilinde yapılması hedeflenmektedir. Uygulama için CI ve CD süreçlerinde Artifactory[5], Jenkins[6], Docker ve Kubernetes araçları kullanılmış, geliştirme ortamından canlı ortama kadar olan tüm akış otomasyon dâhilinde yönetilmiştir. %100 otomatik test kapsaması sağlanan mikroservisler için kaynak kod Jenkins ile otomatik derlenmekte, devreye alım paketi oluşturulmakta, paket Kubernetes aracılığı ile Docker konteynerlere kesinti yaşanmadan orkestra edilerek aktarılabilmektedir. Manuel test edilen mikroservisler için de test sonrası istenildiğinde ve kesintisiz olarak devreye alma yapılabilmektedir.

Tablo 1. Mikroservisler ve otomatik test kapsama oranları.

Mikroservis	Mikroservis Tanımı	Otomatik Test Kapsama Oranı (%)
performance-ws	Gerçekleşme verisi alma	100
sla-validator	Veri doğrulama	100
enricher	Veri zenginleştirme	100
sla-caching-service	Cache veri okuma	100
metric measurement	Metrik ölçme	100
slo-calculator	Süre hesaplama	100
scheduler-service	Zamanlama	100
process-executor	Süreç çalıştırma	100
penalty-calculation	Ceza ücreti hesaplama	100
sls-exp-calculator	SLS hesaplama	100
notification-ws	SLA tanımı alma	*happy path* ≃ 50
notification-service	SLA tanım yönetme	*happy path* ≃ 50
10 adet mikroservis	Arayüz, tanımlama, raporlama, uygulama yönetimi	0

[4] https://cucumber.io/
[5] https://jfrog.com/artifactory/
[6] https://jenkins.io/

3 Öğrenilmiş Dersler

MSA'de teorik olarak mikroservisler kod düzeyinden veritabanı seviyesine kadar birbirinden soyut şekilde oluşturulmalıdır. Proje dâhilinde geliştirici sayısı, veri büyüklüğü ve bütünlüğü, çok fazla kod tabanı (codebase) olmasının getirdiği hata ayıklama zorlukları gibi faktörler mikroservis sayısını belirlemede etkili olmuştur. Operasyon açısından, mevcut sunucu ve insan kaynakları kapasitesi de dikkate alınmıştır. MSA'de her bir mikroservis için ayrı veritabanı oluşturulması önerilse de, uygulama özelinde veri bütünlüğü önemlidir. Bu yüzden fonksiyonel veriler ortak veritabanı üzerinde tutulmuş olup, temel tanım ve uygulama parametreleri ayrı veritabanındadır. Fonksiyonel veriler için MongoDB kullanılmış, temel tanım ve parametreler için Oracle tercih edilmiştir. Uygulama devreye alındıktan sonra entegrasyonda değişiklik olmadan yapılan güncellemelerin diğer uygulamalarla bağımlı şekilde yapılma zorunluluğu ortadan kalkmıştır. Ayrıca her mikroservis kendi içerisinde güncellenip bağımsız olarak devreye alınabilmektedir. Bu şekilde hem uygulamanın kendi içerisinde hem de diğer uygulamalarla bağımlılık açısından elde edilen esneklik ile planlama ve devreye alım kolaylığı sağlanmaktadır. Ayrıca konteyner yapı ile yatay genişleme kolaylıkla sağlanabilmektedir. Birden fazla mikroservisi kapsayan kullanım durumlarını bütünleşik test edebilmek zorlayıcı bir durum olup, bunun için BDD araçları büyük kolaylık sağlamıştır. BDD aracı sayesinde sağlanan test otomasyonu ile %5'inin manuel olarak koşulması bir hafta süren testler, bir saatte ve istenildiği zaman otomatik olarak koşulabilen duruma gelmiştir. Bu, geliştirme sonrası devreye alım sürecinin kısalması açısından büyük bir kazanımdır. Fakat yapılan her geliştirme için otomatik test senaryolarında oluşabilecek güncelleme, kazanımın yanında önemsiz olsa da ek efor oluşturabilmektedir. Arayüz ile alakalı mikroservislerde, test senaryosu hazırlamak zaman açısından yüksek maliyetli olduğu için, test otomasyonu yapılmamıştır. Operasyon açısından MSA'de bir uygulamayı izlemenin daha zor olduğu görülmüş, bunun için özel izleme araçlarına ve daha nitelikli çalışanlara ihtiyaç duyulmuştur. MSA'de uygulama geliştirme ve operasyon altyapısı oluşturmanın monolitik mimariye göre daha zor olduğu ama teknolojilerle uyum sağlandıktan sonra uygulama yaşatma sürecinin daha kolay olduğu görülmüştür.

4 Sonuç

Bu çalışmada MSA ile geliştirilen bir uygulamanın monolitik uygulamalara entegre edilerek, tasarım ve devreye alma süreci anlatılmıştır. Geliştirme aşamasında fonksiyonel olarak ayrıştırılarak modellenen mikroservisler ile birbirinden bağımsız olarak sağlanan geliştirme ve devreye alma esnekliği CD ve CI araçları ile desteklenmiştir. Vaka çalışmasında kullanılan SLA uygulamasının, kurumdaki diğer uygulamalarla çok fazla entegrasyonu mevcut olup, bu entegrasyonların ayrı mikroservis olarak geliştirilmesi uygulamanın kendi içerisinde olan güncellemelerinin esnek devreye alınabilmesini sağlamıştır. Test otomasyonu açısından %100 kapsama sağlanan mikroservisler geliştirme sonrası insan faktörü olmadan yaygınlaştırılabilmiştir. Bundan sonraki süreçte kurum içerisindeki diğer

uygulamaların MSA'e dönüştürülmesi hedeflenmektedir. MSA'in TMForum [8] gibi çerçeveler üzerinde de önemli değişiklik etkisi oluşturacağı düşünülmektedir.

Kaynakça

1. Türk Telekom Yatırımcı İlişkileri, http://www.ttyatirimciiliskileri.com.tr/tr-tr/turk-telekom-grubu/grup-sirketleri/sayfalar/turk-telekom.aspx
2. Fowler, M.: MonolithFirst, https://martinfowler.com/bliki/MonolithFirst.html.
3. Lewis, J., Fowler, M.: Microservices. https://martinfowler.com/articles/microservices.html.
4. Söylemez, M., Tarhan, A.: Mikroservis Mimarisi ve Mimari Faktörleri Üzerine Endüstriyel Bir İnceleme, 12. Ulusal Yazılım Mühendisliği Sempozyumu, İstanbul, (2018).
5. Aytekin, A. I., Macit, Y.: GelİŞlet (DevOps) Yaklaşımında Konteyner Dönüşümü Deneyimi, 12. Ulusal Yazılım Mühendisliği Sempozyumu, İstanbul, (2018).
6. Çıtlık, A., Bağcı, V.H., Turgut, U.O., Güngör, T.: "Bankacılık Alanında Doğal Dil İşleme Destekli Davranış Güdümlü Geliştirme", 8. Ulusal Yazılım Mühendisliği Sempozyumu, Güzelyurt KKTC, (2014).
7. Continuous integration | ThoughtWorks, https://thoughtworks.com/continuous-integration
8. Frameworx homepage 2019 - TM Forum, http://tmforum.org/tm-forum-frameworx-2/

Büyük Ölçekli Ve Kullanımda Olan Projelerde Ana Mimari Değişiklikleri

Karani Kardaş

Havelsan A.Ş., Komuta Kontrol ve Savunma Teknolojileri Direktörlüğü, 06510, Çankaya, Ankara - Türkiye.
kkardas@havelsan.com.tr

Özet. Büyük ölçekli ve kullanıcı sayısı fazla olan canlı yazılım projelerinde teknolojik değişimler başta olmak üzere maliyetlerin azaltılması, ciddi hataların çözümü, ek fonksiyonel istekler gibi birçok nedenlerle büyük mimari değişikliklere ihtiyaç olabilir. Canlı sistemi bozmadan ve kullanıcı alışkanlıklarını değiştirmeden bu büyük değişikliklerin ve yeniliklerin sisteme eklenmesi büyük bir ihtiyaçtır. Bu çalışma büyük ölçekli bir yazılım projesinin birçok modülünde kullanılan Coğrafi Bilgi Sistemleri altyapısının daha teknolojik kütüphaneler kullanılarak değiştirilmesini ve geliştirilmesini adım adım sunar. Sonuç olarak modüller hangi geliştirimin aktif olduğunu bilmeden arayüzler üzerinden CBS kabiliyetlerine erişebilmekte ve ihtiyaçlarını karşılayabilmektedir.

Anahtar Kelimeler: Mimari Değişikliği, Coğrafi Bilgi Sistemleri, NASA WorldWind Uygulaması

Major Architectural Changes in Large Scale Projects

Karani Kardaş

Havelsan A.Ş., Komuta Kontrol ve Savunma Teknolojileri Direktörlüğü, 06510, Çankaya, Ankara - Türkiye.
kkardas@havelsan.com.tr

Abstract. Major architectural changes may be required in large-scale software projects for many reasons such as technological changes, reducing costs, solving important errors, and additional functional requests. Adding these major changes and innovations to the system is a great need without ruining the live system and changing user habits. This study presents a step-by-step operations of changing and developing the Geographic Information Systems infrastructure, which is used in many modules of a large-scale software project, by using more technological libraries. As a result, modules can access and use GIS capabilities through interfaces without knowing which implementation is active.

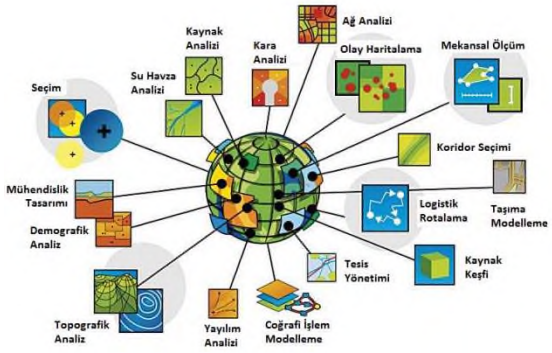
Keywords: Architectural Change, Geographic Information Systems, NASA WorldWind Application.

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

1 Giriş

Başarılı yazılım projeleri bitirilip canlı kullanımına verildikten sonra geliştirilmeye ve iyileştirilmeye ihtiyaç duyarlar. Başarılı projeler daha uzun süre kullanımda kalır ve kullanımda kaldıkça teknolojisi eskir. Canlı, yaşayan sistemin zamanla yeni teknolojilere adaptasyonu önemli ve zor bir süreçtir. Hatta proje büyüklüğü ve kullanıcı sayısı arttıkça bu değişiklik daha risklidir. Canlı kullanımı olan ve kısa süreli aralıklarla sürüm güncellemeleri olan büyük projelerde işleyişi bozmayacak şekilde ve kullanıcı alışkanlıklarını kötü yönde etkilemeyecek şekilde ve hatalara sebep olmadan büyük değişikliklerin yansıtılması önemlidir [1,2].

Coğrafi Bilgi Sistemleri (CBS) coğrafi bilgilerin kullanıcıya gösterildiği, kullanıcının şekil bilgilerini altlık haritalar üzerinde görebileceği, çeşitli değişiklikler yapabileceği, kullanıcı ihtiyaçları doğrultusunda çeşitli analizler yapabileceği bir ortam sunar [3]. Genel olarak veri, insan, yazılım, donanım, personel, analiz ana boyutlarından oluşur. Yazılım sistemlerinde başlıca ihtiyaç duyulan CBS kabiliyetleri; harita açma, raster ve vektör harita gösterimi, katman gösterimi, şekil oluşturma ve değiştirme kabiliyetleri, ölçek / projeksiyon / datum dönüşümleri, harita üzerinde mesafe / alan ölçüm yetenekleri, şekil kesişim analizleri, radar kaplama çizimi ve görünürlük analizleri gibi kabiliyetlerden oluşur. Örnek CBS Kullanımları Şekil 1'de gösterilmiştir.



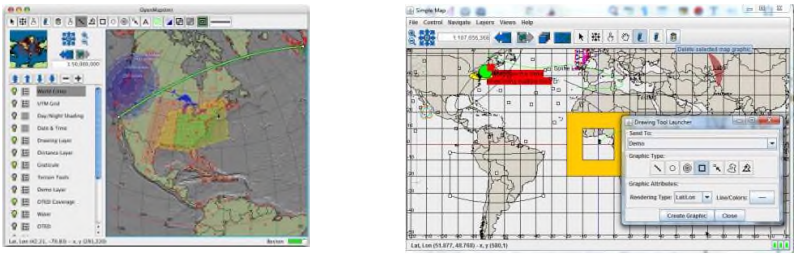
Şekil 1. Örnek Coğrafi Bilgi Sistemleri Kullanımları ([4]'den alıntılanmıştır)

Hemen her alanda teknoljik iyileştirmeler CBS yazılımlarının da kalitesini arttırmaya olanak sağlar. OpenGL [5] teknolojisindeki gelişmeler sayesinde ve üç boyutlu gösterim kabiliyetlerinin artması ile iki boyutlu coğrafik gösterimlerden üç boyutlu gösterimlere geçiş sağlanmıştır. Böylece kullanıcı iki boyutlu analizlere ek olarak üç boyutlu analizler ve görüntülemeler de yapabilir hale gelmiştir.

Literatürde büyük mimari değişiklikleri ile ilgili çalışmalar mevcuttur ancak CBS altyapısının değişimi ile ilgili çalışmalar nadirdir [6, 7]. Bu çalışmada sistematik bir şekilde büyük bir yazılımda en az değişiklik ve yüksek güvenirlikle birçok yazılım modülü tarafından kullanılan iki boyutlu CBS altyapısının üç boyutlu hale getirilişi adım adım gösterilmiştir. Bildiri şu şekilde organize edilmiştir: ikinci bölümde çalışmada kullanılan açık kaynaklı CBS kütüphaneler hakkında bilgiler verilmiştir. Üçüncü bölümde geliştirilen yazılım detaylı bir şekilde adım adım sunulmuştur. Son bölümde geliştirimin artı ve eksi yönleri tartışılmış, sonuçlar ve gelecek çalışmalar değerlendirilmiştir.

2 Kullanılan Coğrafi Bilgi Sistemleri Kütüphaneleri

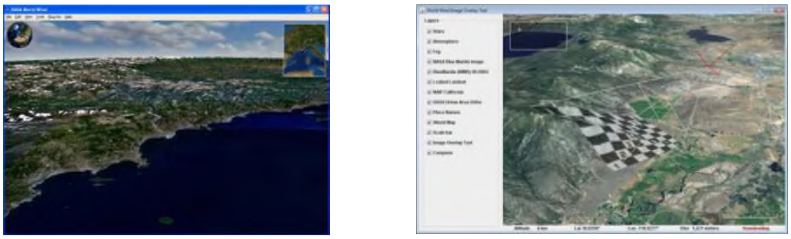
Önerilen çalışmada kullanılan CBS kütüphaneleri OpenMap [8] ve WorldWind [9] açık kaynaklı kütüphaneleridir. OpenMap, Java programlama dili tabanlı ve iki boyutlu bir CBS kütüphanesidir. OpenMap bileşenleri bir çerçeve altyapısı sağlar ve yazılımlara bu bileşenleri ihtiyaç doğrultusunda değiştirme imkanı tanır. Bu bileşenler ile coğrafik bilginin harita üzerinde gösterimi sağlanmış olur. OpenMap uygulamaları Java uygulaması olarak çalışabildiği gibi tarayıcı üzerinden de çalıştırılabilir. Open Geospatial Consortium (OGC) [10] standartlarından etkin raster gösterimini sağlayan Web Map Service (WMS) standardını destekler. OpenMap kütüphanelerinin kullanımını gösteren örnek görüntüler Şekil 2'de gösterilmiştir.



Şekil 2. Örnek OpenMap Uygulama Görüntüleri

OpenMap kütüphanesi farklı projeksiyon yöntemlerini desteklese de üç boyutlu görünüm yeteneğine sahip değildir. Bu da detaylı analizleri sınırlandırmaktadır. OpenMap kütüphanesinin güncel teknolojileri içermemesi, sınırlı OGC standardını destekliyor olması ve üç boyutlu analizlere olanak sağlamaması nedeni ile bu kütüphanenin kullanımı giderek azalmaktadır.

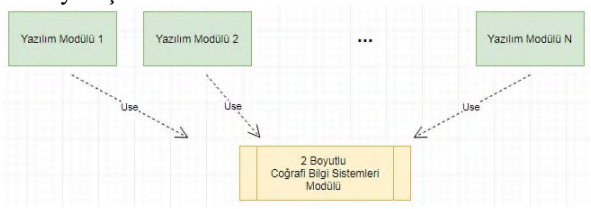
NASA'nın geliştirdiği WorldWind kütüphanesi açık kaynaklı üç boyutlu küre analizi sağlar. OGC standartlarının büyük çoğunluğunu destekler. WorldWind kütüphanesi kolaylıkla ve ihtiyaçlara uygun şekilde geliştirilebilir. Bu kütüphane platform bağımsızdır. Bir başka deyişle Windows, Macintosh, Linux gibi farklı işletim sistemleri üzerinde geliştirme yapılabilir. Ağ ve mobil ortamlar için de yazılım geliştirilmesine olanak tanır. WorldWind ekran kartının sahip olduğu yetenekleri kullanır; böylece günümüz coğrafi bilgi sistemlerinin ihtiyaç duyduğu üç boyutlu görselleştirme işlemini performanslı bir şekilde gerçekleştirir. WorldWind, görsel coğrafi bilgi sistemi geliştirmek isteyen yazılımcılar için çok güçlü, etkileşimli üç boyutlu ve kolay bir ortam sunar [11]. WorldWind kütüphanelerinin kullanımını gösteren örnek görüntüler Şekil 3'te gösterilmiştir.



Şekil 3. Örnek WorldWind Uygulama Görüntüleri

3 Geliştirilen Sistem

Mimarisi bu çalışmada önerilen yöntemle güncellenmesi amaçlanan sistemin mevcut mimarisi Şekil 4'te gösterildiği gibidir. Bir çok modül tarafından kullanılan OpenMap tabanlı iki boyutlu CBS yeteneğine bağlılık şekilden de görüldüğü gibi oldukça yüksektir. Bu mimari yapısıyla CBS Kütüphanesini kullanan modüller CBS değişikliklerinden büyük şekilde etkilenebilir.



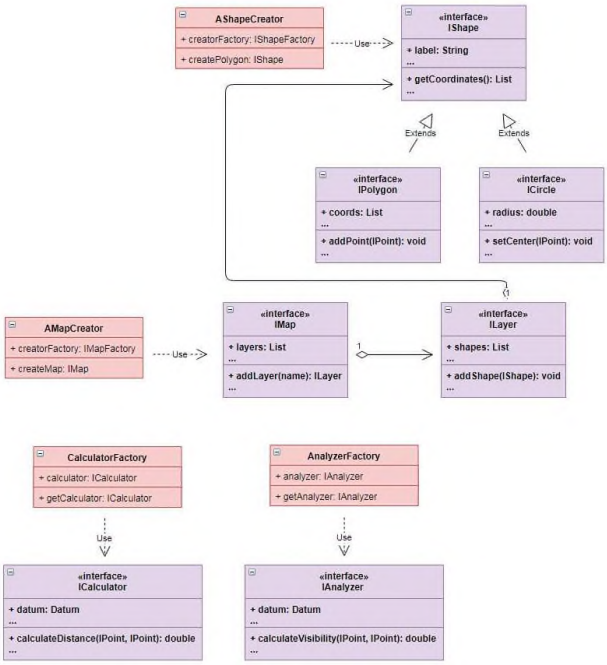
Şekil 4. Sistem Geliştirilme Öncesi Mimari

Mimari değişikliği aşama aşama gerçekleştirilmiştir. İlk aşamada yazılım testleri hazırlanmıştır. Var olan sistemi bozmama amacıyla test sürecine büyük önem verilmiştir. Büyük bir değişiklik yapılacağı için öncelikle var olan sistemi bozmama adına gerekli birim testleri ve senaryo testleri hazırlanmıştır. Böylece sistemin güvenirliği arttırılmıştır. Bu aşamada karşılaşılan en büyük zorluk CBS kullanıcı arayüzü testlerinin yazılmasıdır. Abbot testleri [12] ile bu ihtiyaç kısmi olarak sağlanabilse de bazı CBS analizi tabanlı arayüz testleri için Abbot testleri yetersiz kalmaktadır. Ancak saf CBS hesap ve analizleri için birim testleri yazılmıştır. Coğrafik hesaplamaların doğruluğu MATLAB gibi yazılımların coğrafik hesaplamalarından alınan değerlerle veya İnternet'teki çevrimiçi hesaplamaların yapıldığı sitelerden alınan değerlerle sağlanmıştır.

Çalışmanın ikinci aşamasında CBS yetenekleri için arayüzler tanımlanmış ve bir arayüz çerçevesi (ing. framework) oluşturulmuştur. Fabrika ve soyut fabrika tasarım desenleri başta olmak üzere tasarım desenleri kullanılarak ve "Model-View-Controller" (MVC) mimari desenleri kullanılarak tamamen arayüzler üzerinden CBS ihtiyaçlarını karşılayacak bir çerçeve oluşturulmuştur. Geliştirilen arayüz kümesindeki bazı önemli sınıfların sınıf diyagramı Şekil 5'te sunulmuştur.

Çalışmanın üçüncü aşamasında CBS kabiliyetlerini kullanan tüm modüllerde iyileştirmeler (ing. refactoring) yapılmıştır. Böylece modüllerdeki ve testlerdeki CBS kabiliyetleri kullanımı sadece arayüz üzerinden olabilecektir. Bu işlem sonrasında modüllerde direk kütüphane geliştirimlerine bağlılık ortadan kaldırılmış; CBS modülüne bağlılık sadece CBS arayüz çerçevesi üzerinden olacak hale getirilmiştir. Modüller gerekli arayüzlere erişimleri fabrika sınıfları üzerinden alabilir. Bu fabrika sınıfları çalışma anı parametrelerine göre iki boyutlu CBS kabiliyet kümesi veya üç boyutlu CBS kabiliyet kümesi döndürür. Modüller hangi geliştirimin aktif olduğunu bilmeden arayüzler üzerinden CBS kabiliyetlerine erişir ve ihtiyaçlarını karşılar. Tablo 1'de modüllerden CBS kabiliyetlerine arayüzler üzerinden yapılabilen örnek erişimlerin Java kod örnekleri listelenmiştir.

Çalışmanın dördüncü aşamasında önceki iki boyutlu OpenMap kütüphane kullanımının arayüz çerçevesi üzerinden yapılabilmesi sağlanmıştır. Böylece OpenMap kütüphanelerine direk erişilerek yapılan CBS ihtiyaçları, arayüz üzerinden sağlanır hale getirilmiştir.

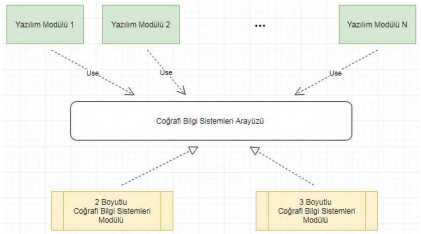


Şekil 5. Uygulamanın Arayüz Setindeki Bazı Önemli Sınıfların Ana Sınıf Diyagramı

Tablo 1. Modüllerin Arayüzler Üzerinden CBS Kabiliyetlerini Kullanım Örnekleri

Harita oluşturma	`IMap myMap = AMapCreator.createMap();`
Katman oluşturma	`ILayer myLayer = myMap.createLayer("Deneme");`
Şekil oluşturma	`IPolygon myPolygon = AShapeCreator.createPolygon(coordinates);`
Coğrafik Hesaplamalar	`ICalculator myCalculator = CalculatorFactory.getCalculator();` `myCalculator.calculateDistance(coord1, coord2);` `myCalculator.calculateAngle(coord1, coord2);`
Analizler	`IAnalyzer myAnalyzer = AnalyzerFactory.getAnalyzer();` `myAnalyzer.calculateVisibility(coord1, coord2);` `boolean result = myAnalyzer.intersects(shape1, shape2);` `myAnalyzer.convertCoordinate();`

Beşinci aşamada ise arayüz sınıflarının üç boyutlu WorldWind kütüphaneleri kullanılarak gerçekleştirimi yapılmıştır. Yazılımın iki boyutlu ve üç boyutlu CBS kütüphaneleri ile birlikte çalışabilirliği çalışma anında derlemeye gerek kalmadan uygulama açılışında kullanıcı seçimi ile gerçekleştirilmiştir. Böylece modüller gerçekleştirim detayından uzak bir şekilde iki boyutlu veya üç boyutlu CBS ihtiyaçlarını karşılar. Çalışma sonrası oluşan mimari Şekil 6'da gösterilmiştir. Tüm yazılım modülleri CBS ihtiyaçlarını arayüz üzerinden karşıladığı için modüllerin olası ileri kütüphane değişikliklerinden etkilenmesi de en aza indirilmiştir.



Şekil 6. Mimari Değişikliği Sonrası Sistemin Genel Görünümü

4 Sonuçlar ve Değerlendirme

Bu çalışmada, büyük ölçekli bir yazılım projesinin birçok modülünde kullanılan CBS altyapısının daha teknolojik kütüphaneler kullanılarak değiştirilmesi sunulmuştur. CBS yeteneklerine erişim için genel bir CBS arayüzü çerçevesi oluşturulmuş ve modüller tarafından CBS yeteneklerine erişim bu arayüz çerçevesi üzerinden sağlanmıştır. Bu arayüz çerçevesinin hem iki boyutlu hem de üç boyutlu gerçekleştirimi yapılmıştır. Tüm bu işlemler yapıldığında çalışan sistemi bozmama adına birim, senaryo, Abbot ve kullanıcı testlerine önem verilerek sistem güvenirliği arttırılmıştır. Sonuç olarak modüller hangi geliştirimin aktif olduğunu bilmeden arayüzler üzerinden CBS kabiliyetlerine erişebilmekte ve ihtiyaçlarını karşılayabilmektedir. Modüllerin kütüphane bağımlılıkları ortadan kaldırılmış olur. Ayrıca çalışmada riskin düşürülmesi için üç boyutlu CBS sisteminin iki boyutlu CBS sistemi ile birlikte çalışılabilmesi sağlanmış; böylece öngörülmeyen bir hata durumunda kullanıcının eski iki boyutlu CBS Kütüphanesi ile işlemine devam edebilmesi sağlanmıştır. İki kütüphane ile birlikte çalışabilirlik ayrıca donanımsal hazırlığa da süre sağlar. Bu CBS değişikliği yeni üç boyutlu teknolojileri kullandığından ekran kartı iyileştirmesi gibi donanımsal değişikliklere de neden olabilir. Henüz donanımı hazır olmayan kullanıcılar eskisi gibi iki boyutlu sistem üzerinden devam edebilir. Üç boyutlu sistemin tamamen iki boyutlu sistemin ihtiyaçlarını karşıladığı kullanıcı kararı ile belirlendiğinde bu değiştirebilme yeteneği kaldırılabilir.

Çalışma sırasında karşılaşılan zorlukların başında teknoloji uyumsuzluğu verilebilir. Üç boyutlu Worldwind kütüphanesi OpenGL bağımlı olduğundan bazı Java Swing uyumsuzlukları yaşanmıştır. Ayrıca kullanıcı etkileşimli CBS işlemleri için geliştirilen ve otomatize edilen testlerde zorluklar yaşanmıştır. Kullanıcı testleri ile bu zorluk aşılmıştır.

Gelecek çalışma olarak, bu mimarinin altyapısal değişikliği planlanmakta ve istemci kısmının inceleştirilerek (ing. thin-client) ağ üzerinden gerçekleştirilmesi hedeflenmektedir.

Kaynakça

1. Shaw, M., & Garlan, D.: Software architecture (Vol. 101). Englewood Cliffs: prentice Hall. (1996)
2. Bass, L., Clements, P., & Kazman, R.: Software architecture in practice. Addison-Wesley Professional. (2003)
3. Clarke, K.: Getting started with GIS. (1999)
4. GIS – The Geographic Approach , https://www.esri.com/news/arcnews/fall07articles/gis-the-geographic-approach.html, Son Erişim 2019/06/05
5. Woo, M., Neider, J., Davis, T., & Shreiner, D.: OpenGL programming guide: the official guide to learning OpenGL, version 1.2. Addison-Wesley Longman Publishing Co., Inc.. (1999)
6. Le, D. M., Behnamghader, P., Garcia, J., Link, D., Shahbazian, A., & Medvidovic, N.: An empirical study of architectural change in open-source software systems. In 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories (pp. 235-245). IEEE. (2015)
7. Bennett, K. H., & Rajlich, V. T.: Software maintenance and evolution: a roadmap. In Proceedings of the Conference on the Future of Software Engineering (pp. 73-87). ACM.(2000)
8. An Open Source Java MappingToolkit, http://openmap-java.org, Son Erişim 2019/06/05
9. NASA WorldWind, https://worldwind.arc.nasa.gov/, Son Erişim 2019/06/05
10. Na, A., & Priest, M.: Open Geospatial Consortium Inc. Wayland, MA, USA, 26. (2007)
11. Kardas, K.: Yükseklik Verilerinin Kullanılabilirliği ve Kalitesi: Endüstriyel Durum Çalışması. VII. Uzaktan Algılama CBS Sempozyumu (UZAL-CBS). (2018)
12. Ruiz, A., & Price, Y. W.: Test-driven gui development with testing and abbot. IEEE Software, 24(3), 51-57. (2007)

Dağıtık Sistemlerin İzlenmesi İçin Bir Yazılım Mimarisi Önerisi

İbrahim Onuralp Yiğit, Dinçer Ünal, Ercan Kılıç, Emre Kılıç,
Murat Demirsoy, Burak Gökmen

Havelsan A.Ş., İstanbul, Türkiye
{iyigit, dunal, ercankilic, emrek, mdemirsoy, bgokmen}@havelsan.com.tr

Özet. Bu bildiride Deniz Kuvvetleri Komutanlığı bünyesinde geliştirilen, su üstü ve su altı platformlarında kullanılan dağıtık sistemlerin izlenmesi için önerilen yazılım mimarisi çözümü anlatılmaktadır. Bu çalışma kapsamında dağıtık sistemin gerçek zamanlı olarak izlenebilmesini sağlayan bir veri analitiği platformu geliştirilmiştir. Dağıtık sistemde oluşan ve oluşması muhtemel anormal durumlar bu veri analitiği platformu ile tespit edilip incelenmektedir. Bu sayede dağıtık sistem ağındaki makinelerin anormal durumlardan en düşük derecede etkilenmesi ya da etkilenmemesi sağlanabilecektir.

Anahtar Kelimeler: Veri Analizi, Veri Görselleme, Yazılım Mimarisi, Dağıtık Sistem İzleme, Elasticsearch, Logstash, Kibana

A Proposal of Software Architecture for Monitoring Distributed Systems

İbrahim Onuralp Yiğit, Dinçer Ünal, Ercan Kılıç, Emre Kılıç,
Murat Demirsoy, Burak Gökmen

Havelsan A.Ş., İstanbul, Turkey
{iyigit, dunal, ercankilic, emrek, mdemirsoy, bgokmen}@havelsan.com.tr

Abstract. This paper proposes a software architecture solution developed for monitoring the distributed systems used in the surface and underwater platforms developed within Turkish Naval Forces. Within the scope of this study, a data analytics platform has been developed that enables real time monitoring of the distributed system. Abnormalities occurring and likely to occur in the distributed system are detected and analyzed with this data analytics platform. Therefore, machines in the distributed system network are not affected or affected to a lesser extent because of abnormal conditions.

Keywords: Data Analysis, Data Visualization, Software Architecture, Distributed System Monitoring, Elasticsearch, Logstash, Kibana

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

1 Giriş

Dağıtık sistemler, kullanıcılara tek bir sistem olarak görünen bağımsız bilgisayarların bir ağ içerisinde birbirleri ile iletişim kurarak çalıştığı bir sistemler bütünüdür [1]. Dağıtık sistemler karmaşık çalışma prensibine sahiptirler. Sistemler üzerinde çalıştırılan uygulamalar birbirleri ile sürekli olarak veri alışverişinde bulunmaktadır. Bu veri alışverisi gerçek zamanlı olarak gerçekleşmekte olup ağda oluşacak gecikme ve senkronizasyon sorunları sistem davranışını olumsuz etkilemektedir. Ayrıca dağıtık sistem üzerinde çalışan uygulamaların tükettiği sistem kaynakları (işlemci, bellek, disk, bant genişliği kullanımı vb.) sistem performansını doğrudan etkilemektedir. Bu etkenlerin yanı sıra dağıtık sistem bünyesindeki makinelerin konfigürasyon (donanım, işletim sistemleri, çalışan uygulamalar vb.) farklılıkları göz önüne alındığında çalışma anında anormal durumlarla karşı karşıya kalınabilir. Bundan dolayı dağıtık sistem bünyesinde yer alan bir ya da birden fazla makinede ortaya çıkacak sorun tüm sistemin olumsuz etkilenmesine sebep olabilmektedir.

Dağıtık sistem içerisindeki makinelerin işlemci, bellek, disk, bant genişliği kullanımı, sistem üzerinde çalışan uygulamaların sağlık durumları gibi veriler çalışma anında periyodik olarak üretilmektedir [2]. Bu veriler gerçek zamanlı olarak izlenerek oluşan ve oluşması olası sistem anomalileri tespit edilebilir. Bu bildiride Deniz Kuvvetleri Komutanlığı bünyesinde geliştirilen, su üstü ve su altı platformlarında kullanılan dağıtık sistemlerin izlenmesi için önerilen yazılım mimarisi çözümü anlatılmaktadır. Bu çalışma kapsamında dağıtık sistemin ve sistem üzerinde çalışan uygulamaların gerçek zamanlı olarak izlenebilmesini sağlayan bir veri analitiği platformu geliştirilmiştir. Dağıtık sistemde oluşan ve oluşması muhtemel anormal durumlar bu veri analitiği platformu ile tespit edilip incelenmektedir. Bu sayede dağıtık sistem ağındaki makinelerin anormal durumlardan en düşük derecede etkilenmesi ya da etkilenmemesi sağlanabilecektir.

Bildirinin bundan sonraki bölümleri şu şekilde düzenlenmiştir: İkinci bölümde yapılan literatür araştırması hakkında bilgi verilmiştir. Üçüncü bölümde geliştirilen yazılım mimarisi çözümü ve veri analitiği platformu ile ilgili detaylar paylaşılmıştır. Dördüncü bölümde geliştirilen veri analitiği platformunun kullanımıyla elde edilen sonuçlarla ilgili değerlendirmelere yer verilmiştir. Son bölümde yapılan çalışma sonucunda gelinen noktanın değerlendirmesi yapılmış ve gelecek dönemde yapılacak çalışmalardan bahsedilmiştir.

2 Literatür Taraması

Günümüzde dağıtık sistemlerin belirlenen metriklere göre gerçek zamanlı olarak izlenmesi ve olası sistem anomalilerinin önceden tespit edilmesi araştırmacıların ilgilendiği önemli konulardır [3]. Mehrotra ve arkadaşları dağıtık sistemin izlenmesi için bir uygulama geliştirme çerçevesi sunmuş, bu çerçeveyi kullanarak belirledikleri sistem ve uygulama metriklerini gerçek zamanlı izleyip sistemin hataya dayanıklılığını arttırmışlardır [4]. Martin ve arkadaşları ise dağıtık sistem verilerinin toplanması ve saklanması için bir yazılım mimarisi önermişlerdir

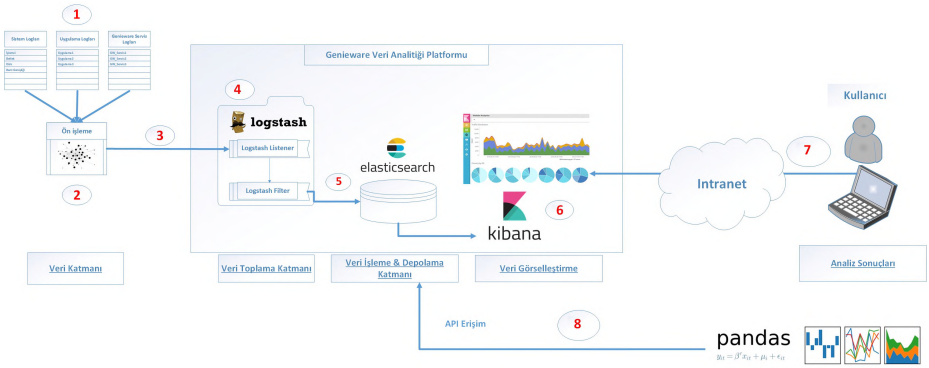
[5]. Önerdikleri yazılım mimarisine uygun uygulama çerçevesi geliştirmişlerdir. Gunter ve arkadaşları da dağıtık sistemlerin performans takibi ve sorunların izlenmesi için NetLogger aracını geliştirmişlerdir [6]. Çalışmalarında NetLogger aracı ile dağıtık sistemden gerçek zamanlı topladıkları kayıt (log) verilerini görselleştirip sistem performansının analiz edilmesini sağlamışlardır. Akdur ve Özdemir, geliştirdikleri yeniden kullanılabilir kayıt izleme kütüphanesi sayesinde radar sistemlerinin ait işlemcilerin anlık yük durumlarını takip edebilmektedirler [7]. Dağıtık sistemlerin gerçek zamanlı izlenmesine ek olarak bu sistemlerde meydana gelebilecek olası anomalilerinin önceden tespit edilebilmesine ilişkin literatürde çalışmalar bulmaktadır. Özer ve arkadaşları yaptıkları çalışmada veri tabanı yönetim sistemleri kayıtlarından oluşturdukları veri seti üzerinde makine öğrenmesi tekniklerini kullanarak hataları tespit etmeye çalışmışlardır [8]. Literatürdeki çalışmalar incelendiğinde, eş zamanlı dağıtık sistem verilerini analiz edip anlık anormal durumları tespit edebilen bir yazılım mimari önerisine rastlanmamıştır.

3 Önerilen Çözüm

Bu çalışma kapsamında Deniz Kuvvetleri Komutanlığı bünyesinde geliştirilen Genesis Middleware (Genieware) [9] arakatman dağıtık sistemi ile entegre çalışacak bir yazılım mimarisi çözümü önerilmektedir. Genieware, içerisinde birçok farklı işlevdeki uygulamayı "yayınla - abone ol" mekanizması ile senkronize bir şekilde çalışmasını sağlayan, deniz savaş yönetim sistemlerinde kullanılan bir arakatman yazılımıdır [10]. Genieware, birçok makine konfigürasyonu (kullanıcı konsolu, merkezi işlem birimi, alt sistem birimi, genel destek birimi ve dosya saklama birimi) üzerinde çalışan uygulamalar arasında haberleşmeyi ve veri paylaşımını sağlamaktadır [11]. Genieware üzerinde çalışan uygulamalar Java, C, C++ gibi farklı programlama dilleri ile geliştirilmiştir. Bu uygulamaların çok farklı işlevleri ve performans gereksinimleri bulunmaktadır. Dağıtık sistem üzerinde uygulamaların yüksek performanslı ve verimli bir şekilde çalışmasını Genieware sağlamaktadır.

Bu çalışma kapsamında dağıtık sistem verilerinin analizi için bir veri analitiği platformu geliştirilmiştir. Dağıtık sistem bünyesindeki makinelerin işlemci, disk, bellek, bant genişliği kullanımı, bu makineler üzerinde çalışan uygulamalar ve bu uygulamaların birlikte çalışmasını sağlayan Genieware servislerinin ürettiği kayıtlar, izlenecek dağıtık sistem verilerini oluşturmaktadır. Geliştirilen veri analitiği platformu kullanılarak dağıtık sistem bünyesindeki makinelerin kullanımı, dağıtık sistem üzerinde çalışan uygulamaların sağlık durumlarının ve performanslarının takibi, sistemin altyapısını oluşturan kritik seviyedeki Genieware servislerinin izlenmesi sırasında oluşan veriler çalışma anında periyodik olarak analiz edilmektedir. Ayrıca, makinelerin ve uygulamaların ürettiği metin tabanlı kayıtlar da geliştirilen platformda toplanıp analiz edilebilmektedir. Toplanan veriler görselleştirilerek platformun sunduğu interaktif kullanıcı arayüzü ve grafikler sayesinde dağıtık sistemdeki sorunların tespit edilmesi kolaylaşmıştır.

Bu bağlamda Şekil 1'de gösterilen yazılım mimarisi tasarlanmıştır. Şekil 1'de aynı zamanda sistemdeki veri akışına ilişkin aşamalar da gösterilmektedir.



Şekil 1. Genieware veri analitiği platformu yazılım mimarisi

Önerilen yazılım mimarisi aşağıdaki beş temel katmandan oluşmaktadır.

a) **Veri Katmanı:** Dağıtık sistem mimarisi içerisindeki her bir makinede bir ajan uygulama bulunmaktadır. Bu ajan uygulama, bulunduğu makinede belirlenen sistem metriklerini toplamaktadır. Bu uygulama topladığı verileri ön işlemeden geçirip ortak bir MySQL veri tabanında depolamaktadır.

b) **Veri Toplama Katmanı:** Logstash, yazılım kayıtlarının ayrıştırma, dönüştürme ve depolama işlemleri gerçekleştiren bir kayıt yönetim altyapısıdır [12]. Bu adımda Logstash, MySQL veri tabanından verileri "Input" bileşeni ile alıp ve bu verileri "Filter" bileşeni ile ön işlemden geçirdikten sonra "Output" bileşeni ile Elasticsearch'e taşımaktadır.

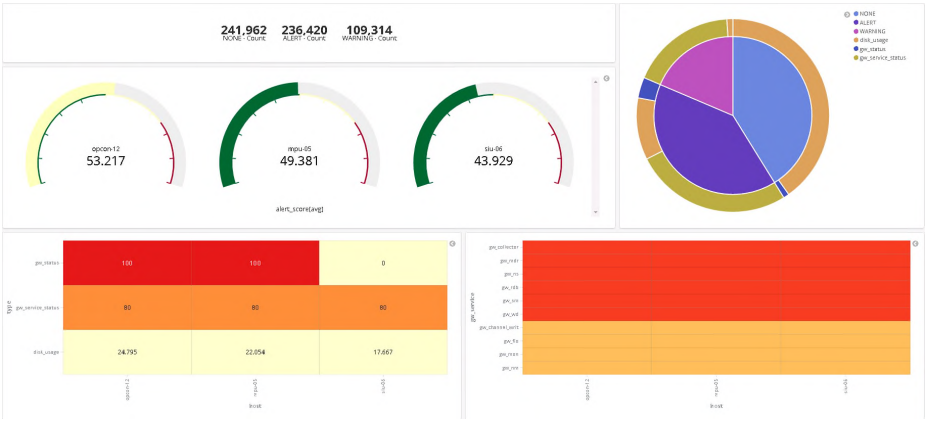
c) **Veri İşleme ve Depolama Katmanı:** Bu aşamada açık kaynaklı, ölçeklenebilir bir tam metin arama ve veri analiz motoru olan Elasticsearch [13] kullanılmaktadır. Elasticsearch aracılığı ile indekslenerek depolanan veriler üzerinden sorgu işlemleri gerçekleştirmektedir. Ayrıca, Elasticsearch desteklediği bir çok API kütüphanesi ile dağıtık sistem verileri üzerinde çeşitli sorgular ile analiz yapılmasına da olanak sağlamaktadır.

d) **Veri Görselleştirme Katmanı:** Veri görselleştirme işleminde veri analizi ve görselleştirme aracı olan Kibana [14] kullanılmaktadır. Bu aşamada, kullanıcılar Kibana aracında hazırlanan interaktif kullanıcı arayüzleri ve grafikler aracılığıyla veri analizi işlemlerini gerçekleştirmektedirler ve sistemde var olan veya oluşması muhtemel sorunları tespit etmektedirler.

4 Değerlendirme

Dağıtık sistem ve sistem üzerinde çalışan uygulamaların ürettiği verilerin bu platform aracılığı ile görselleştirmesi hem geliştiricilerin hem de operasyon ekiplerinin işlerini kolaylaştırmıştır. Bu platform, kullanıcılarına oluşan ve oluşması

muhtemel sorunlarla ilgili analizleri, uyarıları ve alarmları sunmaktadır. Geliştirilen veri analitiği platformunda Şekil 2'deki kullanıcı arayüzü üzerinden dağıtık sistem izlemesi yapılmaktadır.



Şekil 2. Genieware dağıtık sistem izleme aracı kullanıcı arayüzü

Şekil 2'de bir günde toplanan verilere ilişkin analiz çıktısı gösterilmiştir. Sistemde bulunan üç farklı makine türüne ait anormal olarak algılanan durumlar gösterilmiştir. Genieware'e ait bazı uygulamaların düzgün çalışmadığı anlaşılmaktadır. Sistemde gerçekleşen işlem sayısına göre ortaya çıkan hata oranları; bunların hangi uygulamalardan ve hangi sebeplerden kaynaklandığı bu ekran sayesinde gözlemlenmektedir. Ayrıca kullanıcılar, bu sistemde oluşan ve oluşması muhtemel anormal durumlarla ilgili uyarıları zaman serisi grafiklerinde görüntüleyebilmektedirler. Farklı makinelerin ve bu makinelerdeki uygulamaların kayıtları tek bir platformda toplanıp kullanıcı dostu ve anlaşılır grafiklerle temsil edilebilmektedir. Böylece, hata oluştuğunda kayıtların arasında iz sürebilmek kolaylaşmış, hatayı çözmek için harcanan zaman ve maliyet azalmıştır.

5 Sonuç

Bu bildiride Deniz Kuvvetleri Komutanlığı bünyesinde geliştirilen Genieware arakatman yazılımının gerçek zamanlı izlenip dağıtık sistemde aksama durumu yaşanmaması için sistemdeki anomalilerinin analiz edilebileceği bir veri analitiği platformu önerilmiştir. Önerilen platform sayesinde makinelerin işlemci, bellek, disk tüketimi ve kullanım yüzdeleri, ağ trafiği, sistem için kritik seviyedeki uygulamaların durumları ve anormal davranışları gerçek zamanlı takip edilebilmektedir. Ayrıca, önerilen platformla birlikte sistemde ortaya çıkan anomaliler interaktif kullanıcı arayüzü ile hızlı, etkin ve kolay bir şekilde kullanıcıya sunmaktadır. Bu platform kullanılarak anomalilerin tespit edilmesiyle zamanında

önleyici önlemler alınabilecektir. Ayrıca, yapılan çalışmada izlenen dağıtık sistemdeki normal ve anormal durumlarla ilgili veriler toplanıp bir veri kümesinin hazırlanması planlanmaktadır. Hazırlanacak veri kümesi üzerinde makine öğrenmesi tekniklerinden yararlanarak anormal durumların öngörülmesi için çözümler üretilebilir. İlerleyen dönemlerde de bu sistem anomalileri makine öğrenmesi teknikleri ile öngörülüp sistemde bu anomali durumlara karşı önlemler alınması üzerine çalışılacaktır.

Kaynakça

1. Tanenbaum, A. S. & Van Steen, M. *Distributed systems: principles and paradigms* (Prentice-Hall, 2007).
2. Joyce, J., Lomow, G., Slind, K. & Unger, B. Monitoring Distributed Systems. *ACM Trans. Comput. Syst.* **5,** 121–150. ISSN: 0734-2071 (Mar. 1987).
3. Goodloe, A. & Pike, L. Monitoring Distributed Real-Time Systems: A Survey and Future Directions (2010).
4. Mehrotra, R. *et al.* RFDMon: A Real-Time and Fault-Tolerant Distributed System Monitoring Approach. ISSN: ISIS-11-107 (2011).
5. Martin, D. L., Cheyer, A. & Moran, D. B. The Open Agent Architecture: A Framework for Building Distributed Software Systems. *Applied Artificial Intelligence* **13,** 91–128 (1999).
6. Gunter, D., Tierney, B., Crowley, B., Holding, M. & Lee, J. *NetLogger: a toolkit for distributed system performance analysis* in *Proceedings 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (Cat. No.PR00728)* (2000), 267–273.
7. Akdur, D. & Ozdemir, C. *Gerçek zamanlı gömülü sistemlerde yeniden kullanılabilir ve yapılandırılabilir yazılımların kaliteye etkisi: Radar projeleri destek kütüphaneleri* in. **1221** (Sept. 2014), 177–186.
8. Ozer, S., Harsoglu, R., Akdogan, E., Ekinci, Y. & Zorlu, E. *Sistem Yonetimi Hizmetleri icin Sorun Tahmini ve Onleyici Bakim Programi Gelistirilmesi* in *Proceedings of the 11th Turkish National Software Engineering Symposium, Alanya, Turkey, October 18-20, 2017.* (2017), 424–431. `http://ceur-ws.org/Vol-1980/UYMS17\_paper\_33.pdf`.
9. *Bahriye'nin Türk Toplumuna Öncülük Yaptığı Alanlar* `https://www.dzkk.tsk.tr/icerik.php?icerik_id=334&dil=1`.
10. Oksuztepe, K. & Oruk, T. *Deniz Savaş Yönetim Sistemlerinde Yazılım Geliştirme Sürecinde Verimliliğin Artırılması* in *Proceedings SAVTEK2010, Savunma Teknolojileri Kongresi* (2010).
11. Kutlusan, A., Altmidort, N., Oruk, T. & Duman, A. *A combat management system middleware based on CORBA* in *Proceedings DOA'00. International Symposium on Distributed Objects and Applications* (2000), 345–354. doi:`10.1109/DOA.2000.874206`.
12. Logstash. `https://www.elastic.co/products/logstash`.
13. Gormley, C. & Tong, Z. *Elasticsearch: the definitive guide: a distributed real-time search and analytics engine* (" O'Reilly Media, Inc.", 2015).
14. Kibana. `https://www.elastic.co/products/kibana`.

Dağıtık Hizmet Reddi Saldırılarının Tekrarlayan Sinir Ağlarıyla Tespiti

Serhat Caner[1] ve Murat Erten[2]

[1]İzmir Yüksek Teknoloji Enstitüsü, İzmir, Türkiye
[2] Bakırçay Üniversitesi, İzmir, Türkiye
[1] serhatcaner@iyte.edu.tr
[2] murat.erten@bakircay.edu.tr

Özet. Saldırı tespit sistemleri için birçok araştırmacı, makine öğrenmesi yöntemlerini kullanıp bu sistemlerin başarı seviyesini arttırmaya çalışmıştır. Bu araştırmada, Dağıtık Hizmet Reddi (DDoS) ataklarının yapay sinir ağları kullanılarak tespit edilmesi üzerine çalışma yapılmıştır. Bu çalışmada, Tekrarlayan Sinir Ağı (RNN) yapılarının ileri beslemeli sinir ağlarına kıyasla, sıralı saldırı verilerindeki üstünlüğü incelenmiştir. Güncel atak verilerine sahip CICIDS2017 veri seti; Çok Katmanlı Algılayıcılar (MLP), Uzun/Kısa Süreli Bellek (LSTM) ve Kapılı Tekrarlayan Hücre (GRU) ağlarıyla eğitilip test edilmiştir. Çıkan sonuçlara göre RNN'lerin ileri beslemeli ağlara göre, sıralı saldırı verilerinde üstünlüğünü gözükmektedir fakat bu konu üzerine çalışmalar devam etmektedir.

Anahtar Kelimeler: Dağıtık Hizmet Reddi, Tekrarlayan Sinir Ağları, Uzun/Kısa Süreli Bellek, Kapılı Tekrarlayan Hücre

Detection of Distributed Denial of Service Attacks with Recurrent Neural Networks

Serhat Caner[1] and Murat Erten[2]

[1] Izmir Institute of Technology, Izmir, Turkey
[2] Bakırçay University, Izmir, Turkey
[1] serhatcaner@iyte.edu.tr
[2] murat.erten@bakircay.edu.tr

Abstract. For intrusion detection systems, many researchers used machine learning techniques and try to increase success rate. In this research, detection of Distributed Denial of Service (DDoS) attacks has been worked with using artificial intelligence. In this work, superiority of Recurrent Neural Network structures over Feed-forward Neural Networks has been investigated. CICIDS2017 dataset that has up-to-date attack data has been trained and tested with Multilayer Perceptron (MLP), Long-short Term Memory (LSTM), and

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Gated Recurrent Unit (GRU). According to the result, it is seen that RNNs have an advantage over feed-forward networks, but the studies continue on this topic.

Keywords: Distributed Denial of Service, Recurrent Neural Networks, Long-short Term Memory, Gated Recurrent Unit

1 Giriş

Dağıtık Hizmet Reddi (Distributed Denial of Service) adı ile bilinen DDoS atakları-nın[1], ilk kez 1999 yılında gerçekleşmesinden itibaren gitgide popülerliği arttı[2]. Radware 2017-2018 raporuna göre, en çok öne çıkan ataklardan biri olan DDoS, servis sağlayıcı organizasyonlara göre bir numaralı tehdittir[3]. En ağır ağ trafiğine sebep olan SYN saldırılarının (SYN flood) yanı sıra; UDP, HTTP, TCP ve ICMP saldırıları gibi tipleri de olan DDoS atakları, birçok ağ ve sistem için ulaşılabilirliği engelleyen ağ trafiği oluşturmaktadır[4].

Saldırı tespit sistemleri, John Anderson'un 1980'li yayını[5] itibari ile birçok araş-tırmaya konu olmuştur. Bu sistemlerin yaygın iki türü vardır: İmza Tabanlı (Signiture-based) ve Aykırılık Tabanlı (Anomaly-based)[6]. İmza tabanlı atak tespiti, önceden bilinen atakların veri deseninin bir veritabanında saklanıp, oluşan yeni atak ile veri karşılaştırması yapılarak tespit edilmesidir. Bu tespit şekli hızlı olmakla birlik-te, önceden bilinmeyen bir atak çeşidi karşısında savunmasız kalmaktadır. Bunun yanı sıra aykırılık tabanlı atak tespiti, izlenen ağ trafiğindeki olağandışı ağ hareketlerinin tespit edilmesidir. Bu atak tespit şekli, ilk kez karşılaşılan sıfır gün saldırıları gibi saldırılarda daha üstündür. Aykırılık tabanlı tespiti üç ana kategori olacak şekilde sınıflayabiliriz: istatistik tabanlı, bilgi tabanlı ve makine öğrenmesi tabanlı [7].

Makine öğrenmesi tabanlı teknikler, analiz edilen verilerin sınıflandırılmasını sağ-layan modeller oluşturulması ile yapılır. Bu modellerin eğitimi, etiketli verilerin tasar-lanan bu yapıya öğretilmesi şeklinde gerçekleştirilir. Bu tekniklerin esneklik, uyum sağlama, hızlı işlem yapma ve öğrenme gibi faydaları vardır[8]. Yeni veriler ile ça-lışma şeklini değiştirip sınıflandırma yeteneğini farklılaştırabilir. Bu gibi faydaları ile birlikte makine öğrenmesi yöntemleri, saldırı tespit sistemlerinde kullanılmıştır.

Bu makalede, makine öğrenmesinin dallarından biri olan, insan beyninin yapısını örnek alarak geliştirilen yapay sinir ağlarının (Artificial Neural Networks)[9] farklı çeşitlerinin, saldırı tespit sistemi olarak kullanılması hedeflenmiştir. Yapay sinir ağı tiplerinden olan Tekrarlayan Sinir Ağı (Recurrent Neural Network) kısaca RNN, fark-lı alanda başarısı göstermiştir[10,11]. Bu yapının asıl özelliği, İleri Beslemeli Sinir Ağı (Feed-forward Neural Network) tipindeki ağların aksine, bilginin ağ katmanları arasında dolaştırılarak, önceki bilgilerin hatırlanmasını ve bu verilere göre işlem yap-masını sağlamasıdır. Bu sebeple zaman serisi verilerinin kullanımında avantajı vardır. Yapılan çalışmada, RNN tipindeki sinir ağlarının, akış şeklindeki ağ trafiği verisi üzerindeki avantajının incelenmesi amaçlanmıştır. Bu sebeple, RNN tipinde olan Uzun/Kısa Süreli Bellek (Long-short Term Memory - LSTM)[12] ve Kapılı Tekrarla-yan Hücre (Gated Recurrent Units - GRU)[13] ve karşılaştırma için ileri beslemeli sinir ağlarından olan Çok Katmanlı Algılayıcılar (Multilayer Perceptron - MLP)[14], bu çalışmada kullanılmıştır.

Makalenin geri kalan kısmı şu şekilde düzenlenmiştir: 2. Bölümde ilgili çalışmaların incelenmesine yer verilmiştir. 3. Bölümde kullanılacak olan yöntemler incelenmiştir. 4. Bölümde yapılan deneyin sonuçlarına yer verilmiştir. Son olarak da 5. Bölümde çalışma sonuca bağlanmış ve gelecek çalışmalardan bahsedilmiştir.

2 İlgili Çalışmalar

Saldırı tespit sistemleri, birçok araştırmacı tarafından makine öğrenmesi temel alınarak geliştirilmeye çalışıldı. Bu çalışmalarda, geniş bir yelpazede farklı yaklaşımlar sunuldu [15]. Kimi çalışmalar sade yapıdaki makine öğrenmesi algoritmalarından oluşmaktayken, kimi çalışmalar da farklı metotların bir arada kullanımı ile başarı yüzdesinin artmasını hedeflemektedir.

Karataş ve Şahingöz'ün [16] çalışmasında, iki gizli katmanlı (hidden layer) bir sinir ağının ağırlıklarının güncellenmesinde, farklı eğitim fonksiyonlarını denemişlerdir. KDD Cup'99 [17] veri seti kullanılarak yapılan bu çalışmada Levenberg-marquardt optimizasyonu [18] kullanan ağırlık güncelleme fonksiyonunun en iyi sonuç veren yöntem olduğu hakkında çıkarım yapmışlardır. Başka bir çalışmada Abhiroop ve diğ., yazılım tabanlı ağlarda (Software Defined Networks) [19] farklı algoritmaların karşılaştırmalarına yer verilmiştir [20]. Yazarların ulaştığı sonuç; sinir ağlarının, destekçi vektör makinesi (Support Vector Machine)[21] ve Naive Bayes sınıflandırması [22] yöntemlerinden daha üstün olduğudur.

Bu çalışmaların yanı sıra, farklı algoritmaları birleştiren ya da algoritmalara bazı eklemeler yapan ve hata oranını düşürmeyi hedefleyen çalışmalar da bulunmaktadır. Shinde ve Chatterjee'nin yaklaşımında [23], SVM[21] ve C4.5[24] sınıflandırmaları birleştirilerek kullanılmıştır. KDD Cup'99 [17] gibi veri setlerinin tarihi olduğu kanısıyla NSL-KDD veri seti [25] kullanılarak yapılan bu çalışmada başarılı bir sonuç elde edilmiştir. Başka bir çalışmada Igbe ve diğ., iki metodun kombinasyonunu kullanmışlardır [26]. Bu çalışmada insan bağışıklık sisteminden esinlenilerek, Dentritik hücre algoritması [27] ve Negatif seçme algoritması [28] kullanılmıştır. Başka bir makalede de KDD Cup'99 [17] ve NSL-KDD veri seti [25] üzerinde, GRU[13] ve MLP[14] algoritmaların birleştirilerek kullanılması üzerine yapılan çalışmadan bahsedilmiştir [29]. Ayrıca LSTM[12] algoritmasına da yer veren Xu ve diğ. çalışması, bu makalede bahsedilecek çalışmaya en yakın çalışma olmuştur. Sonuçları ise Çift yönlü (Bidirectional) GRU ve MLP birleşiminin en başarılı sonucu verdiği yönündedir.

3 Yapay Sinir Ağı Modelleri

RNN'ler, ileri beslemeli sinir ağlarının geliştirilmiş halidir. MLP gibi bilinen ileri beslemeli ağlarda bilgi tek bir yöne doğru ilerler [14]; giriş katmanı (input layer), gizli katmanlar (hidden layers) ve çıkış katmanından (output layer) geçerek sonuç elde etmemizi sağlarlar. RNN'lerde ise önceki girdilerin bilgisinin de kullanılmasını sağlayan, aynı katmandaki farklı zaman verilerinin birbirlerine aktarılmasını sağlayan sinir ağı bağlantıları vardır. Bu sayede, bir önceki işlenen girdinin bir sonraki girdi

için kıstas olmasını sağlarlar. Geleneksel RNN yapısında, denklemde (1) gösterildiği üzere her bir t anı için x girdi dizisini, h gizli vektör dizisini, σ lineer olmayan fonksiyonu ve W ağırlık matrisini temsil etmektedir.

$$h_t=\sigma(W_{xh}x_t+W_{hh}h_{t-1}) \tag{1}$$

İleri beslemeli ağlar gibi geri yayılım (back propagation) yöntemini kullanan geleneksel RNN'lerde, yapısından dolayı gradyan yok olması (vanishing gradient) denilen sinir ağı eğitimi sırasında gradyanın önceki katmanlarda çok küçük kalmasıyla sinir ağının ağırlıklarının düzgün güncellenmesini engelleyen bir problem ortaya çıkar [30]. Çıkan bu problem uzun vadeli verilerin hatırlanma oranını gitgide azaltır. Bu problemin üstesinden gelmek için araştırmacılar LSTM[12] ve GRU[13] gibi yöntemleri ortaya atmışlardır.

Hochreiter ve Schmidhuber tarafından geliştirilen RNN'lerin özel bir türü olan LSTM, uzun vadeli verileri hatırlamasıyla geleneksel RNN yapısına üstün gelir. LSTM'lerin geleneksel RNN'lerden farklı olarak, bir gizli katman hücresi için 4 farklı değeri vardır. Denklemde (2) gösterildiği üzere *i*, *f*, *c* ve *o* sırası ile girdi kapısı (input gate), unutma kapısı (forget gate), hücre durumu (cell state) ve çıktı kapısını (output gate) temsil eder. Unutma kapısı hangi bilginin atılıp hangi bilginin hücre durumu için kullanılacağına karar verir. Girdi kapısı hangi bilginin hücre durumu içinde saklanacağına karar verir. Çıktı kapısı ise hangi bilginin çıktı olarak verileceğine karar verir. Bu değerler sayesinde ulaşılan sonuç geleneksel yapıdaki uzun vadeli verilerin gerekli görülmesi halinde hatırlanmasını sağlar.

$$\begin{aligned} i_t&=\sigma(W_{xi}x_t+W_{hi}h_{t-1}+W_{ci}c_{t-1})\\ f_t&=\sigma(W_{xf}x_t+W_{hf}h_{t-1}+W_{cf}c_{t-1})\\ c_t&=f_tc_{t-1}+i_ttanh(W_{xc}x_t+W_{hc}h_{t-1})\\ o_t&=\sigma(W_{xo}x_t+W_{ho}h_{t-1}+W_{co}c_t)\\ h_t&=o_ttanh(c_t) \end{aligned} \tag{2}$$

Cho ve diğ. tarafından geliştirilen GRU ise LSTM ile benzerlik göstermektedir fakat içerdiği kapı sayısı LSTM yöntemine göre daha azdır. Hücre durumu değeri bulunmayan GRU'lar girdi kapısı ve unutma kapısı değerlerini güncelleme kapısı (update gate) olarak birleştirirler. Denklemde (3) gösterildiği üzere *r* değeri sıfırlama kapısı (reset gate), *z* değeri güncelleme kapısı ve $\tilde{h}$ değeri olası çıktı değeri olarak temsil edilir. LSTM'e göre daha az parametresi bulunması ile GRU, performans olarak üstünlük elde eder.

$$\begin{aligned} r_t&=\sigma(W_rx_t+U_rh_{t-1})\\ z_t&=\sigma(W_zx_t+U_zh_{t-1})\\ \tilde{h}_t&= tanh(W_hx_t+U(r_th_{t-1})\\ h_t&= (1-z_t)h_{t-1}+z_t\tilde{h}_t \end{aligned} \tag{3}$$

4 Deneysel Çalışmalar

Deneyde, bu alanda çalışma yapan birçok araştırmacı gibi hazır bir veri setinden yararlanılmıştır. KDD Cup'99 veya NSL-KDD gibi veri setleri çok sayıda araştırmada kullanılmıştır. Fakat bu çalışmada, günümüze gelene kadar değişen ve zenginleşen atak çeşitlerinin daha iyi temsilini barındıran, Kanada Siber Güvenlik Enstitüsü (CIC) tarafından geliştirilen CICIDS2017 veri seti [31] kullanılmıştır. Bu veri seti birçok güncel atağı içerirken, çalışmada kullanılan farklı sinir ağlarına girdi olacak 80'den fazla özellik seti içermektedir.

Yapılan çalışmada, sinir ağlarının oluşturulması için Python'un derin öğrenme kütüphanesi olan Keras API [32] kullanılmıştır. Yapay sinir ağı oluşturmayı kolaylaştıran bu arayüz, çalıştığı bilgisayarın ekran kartı üzerinde işlem yapmayı etkili bir şekilde gerçekleştirmektedir. Bununla birlikte; LSTM ve GRU ağlarını, arttırılmış performansta kullanmak için NVIDIA'nın cuDNN yapısı uygulanmıştır [33].

4.1 Çok Katmanlı Algılayıcılar (MLP)

Öncelikle MLP kontrol deneyi olarak, farklı nöron ve farklı katman değerleriyle kullanılmıştır. Keras API'nin sağladığı model özet tablosu ve elde edilen sonuçlar aşağıdaki gibidir.

Tablo 1. MLP 1. model

Model Şekli:		
Katman (tip)	Çıktı Şekli	Parametre Sayısı
dense_1 (Dense)	(None, 160)	12640
dense_2 (Dense)	(None, 80)	12880
dense_3 (Dense)	(None, 30)	2430
dense_4 (Dense)	(None, 15)	465
Doğrulama skoru: 0,981343420669619		

Tablo 2. MLP 2. model

Model Şekli:		
Katman (tip)	Çıktı Şekli	Parametre Sayısı
dense_1 (Dense)	(None, 160)	12640
dense_2 (Dense)	(None, 320)	51520
dense_3 (Dense)	(None, 80)	25680
dense_4 (Dense)	(None, 15)	1215
Doğrulama skoru: 0,9832920249941358		

Tablo 1 ve Tablo 2'de yer alan veriler incelendiğinde, ara katmanlardaki nöron sayısının arttırılması sonuç üzerinde olumlu etkisini göstermiştir. Fakat Tablo 3'de yer alan verilere bakıldığında anlaşılıyor ki, katman sayısının arttırılması sonuç üzerinde olumsuz etki yaratmıştır.

Tablo 3. MLP 3. model

Model Şekli:		
Katman (tip)	Çıktı Şekli	Parametre Sayısı
dense_1 (Dense)	(None, 200)	15800
dense_2 (Dense)	(None, 600)	120600
dense_3 (Dense)	(None, 150)	90150
dense_4 (Dense)	(None, 30)	4530
dense_5 (Dense)	(None, 15)	465
Doğrulama skoru: 0,9794781866534028		

Toplu olarak incelediğimizde MLP için elde edilen en yüksek skorun %98.32 olduğu sonucu çıkmıştır.

4.2 Uzun/Kısa Süreli Bellek (LSTM)

MLP'nin ileri beslemeli yapısına karşı denenen, RNN çeşidi LSTM için sonuçlar aşağıdaki gibidir.

Tablo 4. LSTM 1. model

Model Şekli:		
Katman (tip)	Çıktı Şekli	Parametre Sayısı
cu_dnnlstm_1 (CuDNNLSTM)	(None, 100)	41200
dense_1 (Dense)	(None, 15)	1515
Doğrulama skoru: 0,9794428602515804		

Tablo 5. LSTM 2. model

Model Şekli:		
Katman (tip)	Çıktı Şekli	Parametre Sayısı
cu_dnnlstm_1 (CuDNNLSTM)	(None, 50)	10600
dense_1 (Dense)	(None, 15)	765
Doğrulama skoru: 0,9871609725217116		

Tablo 4 ve Tablo 5'i incelediğimizde tek bir LSTM katmanı ile yapılan deneyde, LSTM hücresi sayısının arttırılması, sonuç üzerinde kötü bir etki yaratıp skorun düşmesine sebep olmuştur.

Tablo 6. LSTM 3. model

Model Şekli:		
Katman (tip)	Çıktı Şekli	Parametre Sayısı
cu_dnnlstm_1 (CuDNNLSTM)	(None, 1, 64)	36864
cu_dnnlstm_2 (CuDNNLSTM)	(None, 1, 64)	33280
cu_dnnlstm_3 (CuDNNLSTM)	(None, 64)	33280
dense_1 (Dense)	(None, 15)	975
Doğrulama skoru: 0,9832524594240949		

Tablo 7. LSTM 4. model

Model Şekli:		
Katman (tip)	Çıktı Şekli	Parametre Sayısı
cu_dnnlstm_1 (CuDNNLSTM)	(None, 1, 32)	14336
cu_dnnlstm_2 (CuDNNLSTM)	(None, 1, 32)	8448
cu_dnnlstm_3 (CuDNNLSTM)	(None, 32)	8448
dense_1 (Dense)	(None, 15)	495
Doğrulama skoru: 0,9856023716733128		

Tablo 6 ve Tablo 7'de çıkan sonuçlar ise birden fazla kullanılan LSTM katmanının sonuç üzerindeki kötü etkisini göstermektedir. Her ne kadar çıkan sonuç diğer verilere kıyasla yüksek olsa da Tablo 5'te gördüğümüz sonuca yaklaşamamıştır. Toplama bakıldığında LSTM için elde edilen en yüksek skorun %98.71 olduğu sonucu çıkmıştır.

4.3 Kapılı Tekrarlayan Hücre (GRU)

LSTM ile kıyaslama yaptığımız RNN'lerin diğer bir çeşidi olan GRU için sonuçlar aşağıdaki gibidir.

Tablo 8. GRU 1. model

Model Şekli:		
Katman (tip)	Çıktı Şekli	Parametre Sayısı
cu_dnngru_1 (CuDNNGRU)	(None, 50)	7950
dense_1 (Dense)	(None, 15)	765
Doğrulama skoru: 0,9842415986751186		

Tablo 9. GRU 2. model

Model Şekli:		
Katman (tip)	Çıktı Şekli	Parametre Sayısı
cu_dnngru_1 (CuDNNGRU)	(None, 1, 32)	10752
cu_dnngru_2 (CuDNNGRU)	(None, 1, 32)	6336
cu_dnngru_3 (CuDNNGRU)	(None, 32)	6336
dense_1 (Dense)	(None, 15)	495
Doğrulama skoru: 0,9870747761012653		

Tablo 8 ve Tablo 9'da elde ettiğimiz sonuçlara göre, genel olarak skor diğer veriler ile karşılaştırıldığında yüksek bulunmuştur. Fakat inceleme sonucu, LSTM yapısından farklı olarak GRU kullanımında katman sayısının arttırılması sonuç üzerinde olumlu etki yaratmıştır. GRU için elde edilen en yüksek skorun %98.70 olduğu sonucu çıkmıştır.

Genel sonuca bakıldığında, GRU ve LSTM arasında büyük bir fark bulunmamaktadır. Bununla birlikte bu iki RNN, MLP yapısından daha iyi bir sonuç vermiştir.

5 Sonuç

Bu çalışmada, CIC tarafından geliştirilen CICIDS2017 veri seti kullanılmıştır. Bu veri seti, benzer çalışmalarda kullanılan KDD Cup'99 ve NSL-KDD veri setlerine göre daha yeni olduğundan, yapılan çalışmada güncel atak tiplerine ayak uyduran saldırı tespit sistemlerinin geliştirilmesine yardımcı olması beklenmiştir. Saldırı tespit sistemleri için, bu çalışmada kullanılması önerilen RNN tiplerinden olan LSTM ve GRU ve bu iki yapay sinir ağının karşılaştırılması için ileri beslemeli sinir ağlarından olan MLP incelenmiştir. Çıkan sonuçlar yukarıda belirtilmiş olup RNN'lerin beklenildiği üzere daha iyi sonuç verdiği kanısına varılmıştır.

İleriki çalışmalarda, bu algoritmaların parametreleri üzerinde daha detaylı denemeler yapılıp; en uygun nöron, hücre ve katman sayılarının bulunması hedeflenmektedir. Çalışmaların sonucunda elde edilen daha kesin verilerle, yeni veri setleri üzerinde denemeler yapılması planlanmaktadır.

Kaynakça

1. Stein, L. D.: World Wide Web Security FAQ. Lincoln D. Stein. (2002)
2. Criscuolo, P. J.: Distributed denial of service: Trin00, tribe flood network, tribe flood network 2000, and stacheldraht ciac-2319 (No. CIAC-2319). California Univ Livermore Radiation Lab. (2000)
3. "Radware Global application & network security report 2017-2018," Technical Report, (2018)
4. Gutnikov, A., Kupreev, O., & Badovskaya, E.: DDoS attacks in Q1 2019. Kaspersky Lab. URL: https://securelist.com/ddos-report-q1-2019/90792 (2019)

5. Anderson, J. P.: Computer security threat monitoring and surveillance. Technical Report, James P. Anderson Company. (1980)
6. McHugh, J., Christie, A., & Allen, J.: Defending yourself: The role of intrusion detection systems. IEEE software, 17(5), 42-51. (2000)
7. Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E.: Anomaly-based network intrusion detection: Techniques, systems and challenges. computers & security, 28(1-2), 18-28. (2009)
8. Ponce, M. C.: Intrusion detection system with artificial intelligence. In FIST Conference-June. (2004)
9. Zurada, J. M.: Introduction to artificial neural systems, vol. 8, St. Paul: West publishing company. (1992)
10. Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., & Wierstra, D.: Draw: A recurrent neural network for image generation. arXiv preprint arXiv:1502.04623. (2015)
11. Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., ... & Ng, A. Y.: Deep speech: Scaling up end-to-end speech recognition. arXiv preprint arXiv:1412.5567. (2014)
12. Hochreiter, S., & Schmidhuber, J.: Long short-term memory. Neural computation, 9(8), 1735-1780. (1997)
13. Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y.: On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259. (2014)
14. Pal, S. K., & Mitra, S.: Multilayer perceptron, fuzzy sets, and classification. IEEE Transactions on neural networks, 3(5), 683-697. (1992)
15. Mirkovic, J., & Reiher, P.: A taxonomy of DDoS attack and DDoS defense mechanisms. ACM SIGCOMM Computer Communication Review, 34(2), 39-53. (2004)
16. Karatas, G., & Sahingoz, O. K.: Neural network based intrusion detection systems with different training functions. In 2018 6th International Symposium on Digital Forensic and Security (ISDFS), pp. 1-6. IEEE. (2018)
17. Cup, K. D. D.: Intrusion detection data set. The UCI KDD Archive Information and Computer Science University of California, Irvine. URL: http://kdd.ics.uci.edu/databases/kddcup99. (1999)
18. Roweis, S.: Levenberg-marquardt optimization. Notes, University Of Toronto. (1996)
19. McKeown, N.: Software-defined networking. INFOCOM keynote talk, 17(2), 30-32. (2009)
20. Abhiroop, T., Babu, S., & Manoj, B. S.: A Machine Learning Approach for Detecting DoS Attacks in SDN Switches. In 2018 Twenty Fourth National Conference on Communications (NCC), pp. 1-6. IEEE. (2018)
21. Cortes, C., & Vapnik, V.: Support-vector networks. Machine learning, 20(3), 273-297. (1995)
22. Lewis, D. D.: Naive (Bayes) at forty: The independence assumption in information retrieval. In European conference on machine learning, pp. 4-15. Springer, Berlin, Heidelberg. (1998)
23. Shinde, P. J., & Chatterjee, M.: A Novel Approach for Classification and Detection of DOS Attacks. In 2018 International Conference on Smart City and Emerging Technology (ICSCET), pp. 1-6. IEEE. (2018)
24. Quinlan, J. R.: C4. 5: programs for machine learning. Elsevier. (2014)
25. Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A.: A detailed analysis of the KDD CUP 99 data set. In 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1-6. IEEE. (2009)

26. Igbe, O., Ajayi, O., & Saadawi, T.: Detecting denial of service attacks using a combination of dendritic cell algorithm and the negative selection algorithm. In 2017 IEEE International Conference on Smart Cloud (SmartCloud), pp. 72-77. IEEE. (2017)
27. Greensmith, J.: The dendritic cell algorithm (Doctoral dissertation, University of Nottingham, UK). (2007)
28. Hofmeyr, S. A., & Forrest, S.: Architecture for an artificial immune system. Evolutionary computation, 8(4), 443-473. (2000)
29. Xu, C., Shen, J., Du, X., & Zhang, F.: An intrusion detection system using a deep neural network with gated recurrent units. IEEE Access, 6, 48697-48707. (2018)
30. Hochreiter, S.: The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6(02), 107-116. (1998)
31. Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A.: Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In ICISSP, pp. 108-116. (2018)
32. Chollet, F.: Keras. (2015)
33. Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., & Shelhamer, E.: cudnn: Efficient primitives for deep learning. arXiv preprint arXiv:1410.0759. (2014)

JSON Sözdizimli Gerçek Zamanlı İletişim Protokolü

Serkan Ayaz[1] ve Semih Bilgen[2]

[1] Yüksek Lisans Öğrencisi, İstanbul Okan Üniversitesi, Akfırat 34959, Türkiye
serkanayazmail@gmail.com
[2] Danışman, İstanbul Okan Üniversitesi, Akfırat 34959, Türkiye
semih.bilgen@okan.edu.tr

Özet. IoT, kritik görev yazılımları, mikroservisler gibi birçok uygulamada gerçek zamanlı ve platformdan bağımsız iletişim altyapılarına ihtiyaç duyulmaktadır. Bu konuda TCP protokolünün Unicast, Multicast ve Broadcast tipi iletişim tipleri yetersiz kalmaktadır. Aracı sunucu üzerinden yapılan veri aktarımlarında ağ trafiği azaldığından veri iletim performansı artmaktadır. Bu çalışmada platformdan bağımsız ağ istemcileri arasında gerçek zamanlı iletişim kurmak için aracı sunucu üzerinden verilerin aktarıldığı metin tabanlı ve JSON sözdizimli JTP (*JSON Transmission Protocol*) adında yeni bir protokol önerilmektedir. Yapılan çalışmada STOMP, XMPP, RESP ve NATS protokolleri incelenerek protokol boyutları ve algoritma performansı, çerçeve boyutları ve algoritma performansı ve insanlar tarafından okunup kodlanabilme özellikleri değerlendirilmiştir. İncelenen protokoller ile JTP protokolü, performans testleri yapılarak karşılaştırılmıştır. Yapılan değerlendirme sonucunda JTP'nin diğer alternatiflerine göre daha fazla işlevsel özelliğe sahip olduğu gösterilmiştir.

Anahtar Kelimeler: Protokol, JSON, Aracı Sunucu, TCP, JTP, STOMP, XMPP, RESP, NATS, Gerçek Zamanlı İletişim

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Real-Time Communication Protocol With JSON Syntax

Serkan Ayaz[1] ve Semih Bilgen[2]

[1] Graduate Student, Istanbul Okan University, Akfırat 34959, Turkey
serkanayazmail@gmail.com
[2] Supervisor, Istanbul Okan University, Akfırat 34959, Turkey
semih.bilgen@okan.edu.tr

Abstract. Real-time and platform-independent communication infrastructures are needed in applications such as Internet of Things (IoT), critical task software, and microservices. In this regard, Unicast, Multicast and Broadcast communication types of the TCP protocol are insufficient. Data transmission performance increases through data transfer from the broker server, as network traffic is reduced. In this study, we propose a new text-based protocol based on JSON syntax, JTP (JSON Transmission Protocol), in which data is transmitted via the broker server to communicate in real time between platform-independent network clients. In this study, STOMP, XMPP, RESP and NATS protocols were examined and protocol dimensions, frame sizes, algorithm performance, human readability and coding levels were evaluated. The examined protocols and JTP protocol were compared via tests in terms of functional properties. It has been shown that JTP features a higher number of functional properties in comparison to alternatives.

Keywords: Protocol, JSON, Broker Server, TCP, JTP, STOMP, XMPP, RESP, NATS, Real-Time Communication

1 Giriş

İletişim protokolleri bir mesajı veya veriyi diğer bir uç noktaya güvenli ve tutarlı bir şekilde ulaştırmak amacıyla kullanılır. Ancak pratikte bir verinin sadece diğer noktaya iletilmesi yeterli değildir. Veri iletiminin yanısıra iletim kontrollerini de üstlenecek bir yapıya ihtiyaç duyulmaktadır. Bu tip iletişim sistemleri, kurumsal uygulamalarda veri kaybına bağlı hata toleransı sıfıra yakın, kritik görevler üstlenen sunucular arasında iletişimin sağlanması, çok sayıdaki istemciden (IoT cihazları, loglayıcı vb.) verinin toplaması, aynı anda birden fazla hedefe gerçek zamanlı veri aktarımının sağlanmasında, mikroservis [1] tabanlı altyapılardaki servisler arası iletişimin sağlanması gibi amaçlarla kullanılmaktadır. İletişim protokolleri ikili (*binary*) ve metin tabanlı olmak üzere ikiye ayrılır. Metin tabanlı protokollerin programlama dillerinden ve işletim sistemlerinden bağımsız olmaları tüm platformlar arası iletişim durumlarında kullanılmalarına olanak tanımaktadır.

Aracı sunucu (*broker server*), kendisine bağlanan istemciler arasında veri yönlendirmesi yaparak istemcilerin birbiriyle haberleşmesini sağlayan sunucu yazılımıdır. Bir verinin birçok uç noktaya iletilmesinde TCP/IP protokolündeki Unicast, Multicast ve Broadcast'de karşılaşılan problemler aracı sunucu ile çözülmektedir. Ayrıca aracı sunucu ile TCP/IP'nin bir özelliği olan gerçek zamanlı iletişim de desteklenmektedir.

Bu çalışmada, önce, piyasada kullanılan aracı sunucusuna sahip metin tabanlı protokollerin özellikleri incelenerek avantaj ve dezavantajları araştırılmıştır. Araştırma sonucunda genel olarak protokollerin iletişim için yüksek veri boyutu gerektirmesi, kompleks iletişim tiplerini desteklememesi, protokol sözdizimlerinin insanlar tarafından okunup yazılamaması gibi dezavantajlarının olduğu görülmüştür. Çalışmada daha sonra bu temel dezavantajları ortadan kaldıran veya en aza indiren ve bir aracı sunucuyla çalışarak aşağıdaki özellikleri içeren gerçek zamanlı iletişim için kullanılmak üzere JSON (JavaScript Object Notation) [2] sözdizimli yeni bir protokol önerilecektir. Bu protokol, metnin devamında JTP (JSON Transmission Protocol) olarak anılacaktır. JTP protokolünün işlevsel özellikleri şunlardır:

- Gerçek zamanlı çok hedefli iletişim desteği
- Platform bağımsız veri iletimi
- Yayımcı-abone tipi iletişim desteği
- Arayan-aranan tipi iletişim desteği
- Yük dengeleyicili kuyruk desteği
- Süre limitli veri iletimi desteği
- QoS (Quality of Service) iletim denetimi desteği
- Geriye dönük mesajlara erişim desteği
- Protokolün genişletilebilir olması
- Verilerin gerçek zamanlı sıkıştırılarak iletimi
- Kimlik doğrulama desteği
- İletişim tipi ve konu bazlı yetkilendirme
- İzleme desteği

2 Literatür Araştırması

2.1 Araştırma Kriterlerinin Belirlenmesi

Araştırma kapsamının belirlenmesinde aşağıdaki kriterler göz önünde bulundurulmuştur:

- Metin tabanlı veri iletimi
- Aracı sunucu üzerinden haberleşmesi
- Gerçek zamanlı iletişim

2.2 Protokolleri Karşılaştırma Kriterleri

Her protokolün çeşitli avantaj ve dezavantajları bulunmaktadır. Mesajlar ağ üzerinde bir uçtan diğer bir uca aktarılırken protokol ve çerçeve gibi ek veriler eklenerek aktarıldığından protokol ve çerçeve verilerinin boyutu iletişim performansını etkilemektedir [3]. Ayrıca protokol ve çerçeve verilerinin kodlama ve çözümleme işleminde kullanılan algoritmanın karmaşıklığı işlemci kaynaklarını tüketmesi nedeniyle iletişim performansını etkilemektedir [4]. Protokol ve çerçevenin kodlama ve çözme işleminin insanlar tarafından yapıldığında protokol ve çerçevede kullanılan harf ve sayıların dışındaki tüm karakter tiplerinin toplam sayı değeri insanlar tarafından anlaşılabilirliğin ölçütü olarak belirlenmiştir [5]. Bu çalışmanın kapsamına yönelik aşağıdaki karşılaştırma kriterleri belirlenmiştir.

- Protokol boyutu
- Protokol kodlama ve çözümlemede algoritma performansı
- Çerçeve boyutu
- Çerçeve kodlama ve çözümlemede algoritma performansı
- İnsanlar tarafından anlaşılabilmesi ve yazılabilmesi

2.3 Araştırılan Protokoller

Bölüm 2.1'de belirtilen araştırma kriterlerine göre bu çalışmada aşağıdaki protokoller araştırılmıştır. Bu kriterlere uymayan diğer protokoller bu çalışmanın kapsamı dışındadır. Araştırma kriterlerinin bir kısmını karşılayıp bir kısmını karşılamayan protokoller de mevcuttur ve bunlar da kapsam dışı bırakılmıştır.

- STOMP
- XMPP
- RESP
- NATS

Protokol karşılaştırmamızın sonuçları, Çizelge 1'de özetlenmiştir.

3 Yeni Protokol Önerisi

Bölüm 1'de verilen işlevsel özellikleri barındıran platfordan bağımsız ağ istemcileri arasında gerçek zamanlı iletişim kurmak için aracı sunucu üzerinden verilerin aktarıldığı metin tabanlı ve JSON sözdizimli yeni bir protokol önerilmektedir.

3.1 Karşılaştırma Kriterlerine Göre Protokolün Testi

Araştırılan protokollerin karşılaştırma kriterlerine göre protokol testleri, araştırılan protokollerin ve JTP protokolünün desteklediği yayımcı-abone tipi iletişimin yayınlama protokolü örneği üzerinden yapılmıştır. JTP protokolünün testi için aşağıdaki JTP protokol örneği kullanılmıştır.

```
{"Topics":["konular/konu1"],"Payload":"Merhaba Dü-
nya"}\r\n
```

Tablo 1. JTP protokolünün diğer araştırılan protokollerle karşılaştırılması

Kriterler	STOMP	XMPP	RESP	NATS	JTP
Çerçeve Boyutu	2 bayt	19 bayt	2 bayt	2 bayt	2 bayt
Çerçeve Kodlama	O(1)	O(1)	O(1)	O(1)	O(1)
Çerçeve Çözümleme	O(n)	O(n)	O(n)	O(n)	O(n)
Protokol Boyutu	60 bayt	520 bayt	22 bayt	22 bayt	41 bayt (14 bayt)
Protokol Kodlama	O(1)	O(1)	O(1)	O(1)	O(1)
Protokol Çözümleme	O(n)	O(n)	O(n)	O(n)	O(n)
İnsanlar Tarafından Kodlanabilme Değeri	6 veri tipi	9 veri tipi	4 veri tipi	3 veri tipi	10 veri tipi (7 veri tipi)

3.2 İşlevsel Özelliklere Göre Protokolün Testi

Protokoller, belirli sözdizimlerinin biraraya getirilmesiyle protokolün oluşturulma amacına uygun olarak birçok bildirim, komut ve direktif belirtimi yapılabilmektedir. Araştırılan protokollerin oluşturdukları komut setleri yardımıyla destekledikleri özellikler Çizelge 2'de verilmiştir ve JTP protokolünün işlevsel özellikleriyle karşılaştırılması yapılmıştır. STOMP protokolü, basit komutsal iletişim yapabilmek için tasarlandığından işlevsel özellik sayısı bakımından en az işlevsel özelliğe sahip olduğu görülmüştür. XMPP protokolünün genişletilebilir özelliği sayesinde birçok işlevsel özellik sonradan eklenmiştir ve yeni eklenen protokol ekleri standartlaştırılmıştır. RESP protokolü, Redis sunucunun özelliklerini kullanmak

amacıyla tasarlanmıştır. NATS protokolü, yayımcı-abone tipi iletişim işin tasarlanmıştır ancak bunun dışında Çizelge 2'de belirtilen işlevsel özelliklere de sahiptir. Ayrıca araştırılan protokoller arasında en az sayıda komut sayısına sahip olduğu görülmüştür.

Tablo 2. JTP protokolünün işlevsellik yönünden diğer araştırılan protokollerle karşılaştırılması

İşlevsel Özellik	**STOMP**	**XMPP**	**RESP**	**NATS**	**JTP**
Gerçek zamanlı çok hedefli iletişim desteği	Var	Var	Var	Var	Var
Platform bağımsız veri iletimi	Var	Var	Var	Var	Var
Yayımcı-abone tipi iletişim desteği	Var	Var [6]	Var	Var	Var
Arayan-aranan tipi iletişim desteği	Yok	Var [7]	Yok	Yok	Var
Yük dengeleyicili kuyruk desteği	Yok	Var [8]	Var	Var	Var
Süre limitli veri iletimi desteği	Yok	Yok	Var	Yok	Var
QoS (Quality of Service) iletim denetimi desteği	Yok	Var [9]	Yok	Yok	Var
Kalp atışı desteği	Var	Yok	Yok	Var	Var
Protokolün genişletilebilir olması	Yok	Var	Yok	Yok	Var
Verilerin gerçek zamanlı sıkıştırılarak iletimi	Yok	Var [10]	Yok	Yok	Var
Kimlik doğrulama desteği	Yok	Var [11]	Var	Var	Var
İzleme desteği	Yok	Yok	Var	Var	Var

4 Sonuç

JTP adında JSON sözdizimi ile formatlanmış aracı sunucu üzerinden gerçek zamanlı iletişim için kullanılan yeni bir protokol önerilmiştir. Bu protokolün kurumsal uygulamalarda, IoT projelerinde, gerçek zamanlı iletişim gereksinimini karşılamak ve tüm platformlar arasında iletişim kurmak için uygun olduğu gösterilmiştir. Araştırılan protokollerden işlevsel özellikleri bakımından daha fazla özelliğe sahip bir protokoldür. Ancak insanlar tarafından okunabilme ve kodlanabilme özelliği bakımından iyi seviyede olmadığı değerlendirilmektedir. JTP protokolünün desteklemediği ancak literatürde kabul görmüş iletişim tipleri araştırılarak JTP protokolünün geliştirilmesi sağlanabilir. Ayrıca JTP protokolünün kullandığı standart model JSON yerine çalışmanın kapsamı dışında olan daha hızlı kodlama ve çözümleme performansı sunan sözdizimleri kullanılırsa daha da performanslı hale getirilebilecektir. Aracı sunucu ile uç birimler arasında TLS protokolüyle veriler kriptolanarak daha güvenli bir iletişim gerçekleştirilebilecektir.

Kaynakça

1. Namiot, D., Sneps-Sneppe, M.: On Micro-services Architecture, International Journal of Open Information Technologies, 2(9), 2307-8162 (2014).
2. Introducing JSON, https://www.json.org, son erişim 2018/04/23.
3. Bianchi, G.: Performance analysis of the IEEE 802.11 distributed coordination function, IEEE Journal on Selected Areas in Communications, Vol: 18 (2000).
4. Sitaraman, M., Kulczycki, G., Krone, J., Ogden, W.F., Reddy, A.L.N.: Performance specification of software components, New York: ACM (2001).
5. Friedman, S.L., Klivington, K.A., Petersan, R.W.: The Brain, Cognition and Education, Cambridge: Academic Press Inc (1986).
6. Publish-Subscribe, https://xmpp.org/extensions/xep-0060.html, son erişim 2018/04/23.
7. Jingle Session Transfer, https://xmpp.org/extensions/xep-0251.html, son erişim 2018/04/23.
8. Workgroup Queues, https://xmpp.org/extensions/xep-0142.html, son erişim 2018/04/23.
9. Quality of Service, https://xmpp.org/extensions/inbox/qos.html, son erişim 2018/04/23.
10. Message Archiving, https://xmpp.org/extensions/xep-0136.html, son erişim 2018/04/23.
11. Stream Compression, https://xmpp.org/extensions/xep-0138.html, son erişim 2018/04/23.

Sağlık Sistemleri için IoT & Akıllı Sözleşme Kuralı Tabanlı Güvenli İletişim Tasarısı

Aslı Kul[1], Eda Nur Azin[1],Oğulcan Özdemir[1], Serap Sahin[1]
[1]Izmir Institute of Technology, Computer Engineering, Turkey
{aslikul, edaazin, ogulcanozdemir, serapsahin}@iyte.edu.tr

Özet. Nesnelerin İnterneti (IoT) gelişimini sürdüren yeni bir teknolojidir. IoT sensör cihazlarının kullanım amaçlarından biri de kritik bir alan olan tıbbi sağlık sistemlerinde hasta koşullarını izlemektir. Tıbbi cihazların, sensörlerin, uygulamaların entegrasyonu, dijital servisler ve ilgili verilerin gizliliği, doğruluğu ve kullanılabilirliği açısından kritik öneme sahiptir. Bu bildiride, IoT tabanlı sağlık sistemi yapısı tasarlanmış ve çeşitli saldırı türlerine karşı sistemin kullanılabilirliği test edilmiş Bockchain temelli akıllı sözleşme yapısının IoT sistemlerine entegre edilmesiyle sistemin davranışları Calvin ve Etherium Frameworkleri kullanılarak simüle edilmeye çalışılmıştır.

Anahtar Kelimeler*:* Nesnelerin interneti, güvenlik, gizlilik, blok zinciri, akıllı sözleşmeler, very gizliliği, sensor iletişimi, sağlık, ataklar.

IoT - Smart Contract Rule Based Secure Communication Scheme for Healthcare System

Aslı Kul[1], Eda Nur Azin[1],Oğulcan Özdemir[1], Serap Sahin[1]
[1]Izmir Institute of Technology, Computer Engineering, Turkey
{aslikul, edaazin, ogulcanozdemir, serapsahin}@iyte.edu.tr

Abstract. The Internet of Things (IoT) is a new technology that its expansion increases. One of the using purposes of sensor devices is to monitor patient conditions in a critical area as medical healthcare systems. The integration of medical devices, sensors, applications have critical importance from the point of privacy and correctness and availability of digital services and related data. In this paper, an IoT backbone healthcare system was designed and tested its service availability with against frequent security attack types and a Blockchain based smart contract structure is also integrated with IoT systems and behaviors of whole system were tried to simulate by Calvin and Etherium Frameworks.

Keywords*:* Internet of things, security, privacy, blockchain, smart contracts, data privacy, sensor communication, medical, healthcare, attacks.

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

1 Introduction

The Internet of Things is a technology that consists of millions of physical devices worldwide to connect and transmit data to each other through the Internet. It is expected to increase its expansion to be usable in critical areas such as healthcare system or medical industry. Currently, sensor devices based on patient monitoring systems are being used in hospitals. Making these devices cooperation with the Internet of Things (IoT) devices, provides controlling strength and brings advantages of using IoT which are easy monitoring, sharing, and increased usability of collecting data to analyses. Besides that, healthcare purposes IoT system devices contain important patient data with sensitive contents. While the IoT creates a more effective and useful smart world, the work on its security and privacy has increased importance due to the fact that they are designed as lightweight devices and current security approaches are not enough for it.To provide secure IoT systems, different level security measures such as using encryption, secure protocols etc. are used but to manage IoT devices using central authority is one of the other main important problems.

On the other side, Blockchain (BC) is the other popular research area and it consists of time-stamped series of immutable records of data that is managed by the cluster of computers not owned by any single entity. Those key attributes of Blockchain; Peer-To-Peer, Distributed, Cryptographically Secured, Add-Only, Consensus makes it better than traditional approaches. Researches are going on IoT security with the idea that "Integrating Blockchain technology with IoT" can eliminate some security deficiencies.

In this paper, our aim is to provide a secure and smart IoT system for a hospital with integration of Blockchain technology. For this purpose, we design a schema and develop a system using the Digital Certificate scheme and smart contract structure. To prove the system, we develop a simulation environment and test it for some possible attacks.

The paper is organized in five sections: In Section 2, the proposed solution is described to offer security and privacy in IoT based medical industry. Then Section 3 presents implementation results and analysis. In Section 4, some related projects with the security of IoT healthcare systems and their solutions are examined. Section 5 purposes possible future security needs, challenges and perspectives in IoT based healthcare system to conclude in this project paper.

2 Proposed System

The medical purpose secure IOT environment is created for a hospital in this project. The system is built upon 3-layer architecture from bottom to up as shown in Figure 1.

1. **IOT Layer:** In the IoT layer, there are medical IOT devices which consist of EEG, EMG, ECG, glucose monitoring (CGM), insulin pen, temperature sensors, etc. They just collect data and send these data to collectors.

2. **Collector Layer:** Collectors are head clusters. They collect data from devices and defined for each patient in a room. Collectors are used by an Id card for each patient. Data for each patient is signed by Digital Signature Algorithm- DSA when transferring collected data from the collectors to data storage at the application layer.
3. **Application Layer** (Cloud, Db, Web Server, etc)**:** The application layer hosts top-level devices for the medical data storage in cloud or web service. Also besides it gives service to users as client agents with mobile phones, tablets, and personal computers.
4. **Certification Authority:** The certification authority fulfills identification requirement with Public Key Infrastructure-PKI scheme [2] in the system which will be using secure communication between devices in the network. Also ensuring safe initialization vector for the smart contract deployment, which binding identity of authorized system user to smart contract base administration properties.
5. **Ethereum - Smart Contracts:** The smart contracts provide safe and distributed control over data flow in the network with PKI scheme.
   Moreover, three layers share management, authentication, authorization, access control and secure communication properties [1] over the Calvin framework

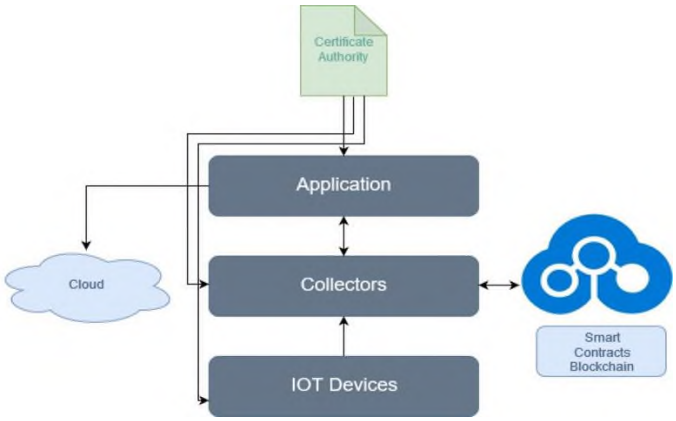


Fig. 1. General components of the proposed solution.

2.1 System Design

Scenerio. In this solution every device and users in the network have public certificates used to identify their role and capabilities in the system. Devices and all entities have the pre-installed certificate before deploying in the system signed by Certificate Authority-CA, which are verified by the ministry of health.

All employees of hospital can use this system by registering to the system and they have id card that enabled by PKI scheme as built in. Each device is defined with related rules to a collector by admin. After the initialization of the device the hospitalization of patient, his/ her admission and verification and registration on the system is done by

authorized person. Health data of patients is collected via IoT devices, and send to the collectors.

A smart contract holds rules for each specific device to verify received data content of IoT devices whether having permission to send those types of data to upper layers or not.The smart contracts are distributed to each collector and application layer devices, so all of them have replicas of it. With a smart contract, the device data structure is validated based on the sender device or group id of the sender certificate. IoT layer devices only send data and do not hold smart contract information. If collectors verified the data according to define rules, it sends them to the database to record otherwise it does not accept it. As shown at Figure 2. While data is sending to the cloud it is signed with private key of patient, thus the identification of the source of received data is provided.

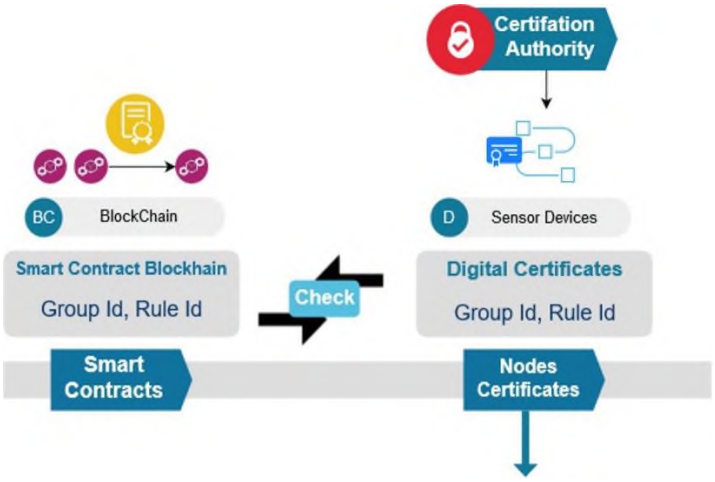


Fig. 2. Group & Rule ID checking while sending data from devices to collectors

Use Cases Requirements. The following use cases and requirements are defined to clearly explain proposed solution.

Actors: IoT Device, Collectors, Certificate Authority, Doctors, Nurses, Patients, Ministry of Health

1. **Rule Policy Definition:** In proposed solution rules are placed in a smart contract. They define which device group can send which type of data. For this purpose each rule is defined as a unique Id and stored as a table entity in the cloud, each contract contains rules against a specific device or group of devices.
2. **Device Registration:** The IoT, collector, and application layer devices are registered to the system to be able to collect and send patients' data from IoT devices and send these data from IoT devices to the collectors.
3. **Patient Registration:** The system gives Id to the hospitalized patient.The given room or defined department for the patient such as the intensive care unit or the

others are recorded to the system as a department, room and bed Id and associated with each other, together with the patient Id and device's Id.

4. **Smart Contract Installation:** The initial chain is started by the authorized system user (admin) with the deployment of rule manager that is a smart contract. Each device has rule policy either dependent for its id or affiliated subgroups such as group, room or floor id. The smart contract also known as a rule manager.the smart contract and each rule contract is only deployed by admin, rule manager smart contracts have a unique feature.They hold and call the registered rule contracts.The rule contracts are specific for their purpose and every rule control against deployed contracts are added as a log to the transaction records.
5. **Certificate Distribution:** Certificates are distributed while registration phase alongside with smart contract client node installation. They contain the necessary extension fields for identification and rule control management and all nodes use certificates to establish a secure communication channel. HTTPS underlying SSL/TLS handshake protocol, the Ministry of health verifies the trustability of the certification authority. In addition, the group ids in the smart contracts and certifications meet the id specifications in x509 standards.
6. **Get Data from Patient:** The main reason to collect data with IoT from patients is to improve the diagnosis and treatment process. The proposed solution implementation combine IoT device data signed with patients' smart identification card which has a built-in PKI scheme [2] with patient's previous data.
7. **Data Collection from Devices*:*** In this solution, data are collected from the IoT devices then send to the collectors by using secure protocol that meets TLS requirements. After successful TLS handshake, device group ID is extracted from IoT certificate by collector and used to find smart contract that holds rules for specific device group. If the data content is approved, the data are sent to the cloud to save data to the database over a secure channel by using the collector certificate.
8. **Save Data to DB:** For the data collected from patients, the hash value is generated by the collectors. Private keys on patients' smart cards are used to sign the hash of the data and then it is sent to the server to save data with their hashes (digital signatures). The database does not accept the data without the hash value of it and check the integrity and identity of the data by verification digital signature process.
9. **Access Patient Data:** The system use a secure communication channel to take the patient's data from the cloud server.

Securiy Requirements And Solutions. For proposed solution in order to offer authentication and identification requirements of the system, the SSL/TLS certificate on PKI scheme was used. In a network, transactions cannot take place without the capacity to digitally identify people and machines in a reliable way. Public Key Infrastructure (PKI) is responsible for offering services required for establishing trusted digital communications [3]. Figure 3 shows that scheme.

In our project rule-based data transformation was set up between layers with using certificates and X509 infrastructure. Certificates include device id, group id for different device types in their extension fields. Each Device id consists of 96 bits electronic product code. Grup id defines device types and collectors so the system is protected against false data injection. All Ids are queried from Rule Authority.

Communication security between all layers such as IoT to collectors and application is provided by the HTTPs protocol with underlying SSL/TLS certificates. Https provides secure communication and confidentiality at a network. Data is sent by HTTPS that is secured via Transport Layer Security protocol (TLS), which provides three key layers of protection.

1. **Encryption:** Encrypting the exchanged data to keep it secure from eavesdroppers. That means that while the user is accessing network, nobody can "listen" to the conversation.
2. **Data integrity:** Data cannot be modified or corrupted during transfer, intentionally or otherwise, without being detected.
3. **Authentication:** Proves that users communicate on network. It protects against men-in-the-middle-attacks and builds user trust, which translates into other business benefits.

Attacks and Prevention. Some IOT devices and IOT network of attacks that may be encountered are listed below. Through the use of certificate authentication and Rule-based approach those types of attacks are prevented for all layers in this system. At IoT layer, false data injection, sinkhole attack, and reply attacks are prevented by rule authority system underlying smart contract infrastructure [12]. Communication channels security is providing by HTTPs underlying the SSL/TLS certificate scheme.

1. **Spoofing Attack:** In the IoT sensor network, IP spoofing is a common way to use while attacking the network. In this project requester identification from a unique X509 based certificate system protects the IoT network from this type of attack.
2. **Node Capture Attacks*:*** Wireless Device Networks may be assortment of sensors with restricted resources that collaborate so as to attain a standard goal. The proposed system has a smart contract based rule control on data over the network which distributed to each collector. Once data arrived to the collector, smart contract id is extracted from sender certificate and data is checked with appropriate smart contract.
3. **Sinkhole Attack:** Sinkhole attack is a corporate executive attack that was an entrant compromise a node within the network and launches an attack. In proposed system sinkholes are named as collectors. Even if one of the collectors is compromised, system also has rule policy control at the application layer.
4. **Replay Attack:** A replay attack is an attack on victimization of the safety protocol. The proposed system does not allow processing of this type of unlegitimate requests by the checking of timestamp within each request.
5. **False Data Injection*:*** Due to their comparatively tiny sizes and unattended operations; sensing of IoT nodes has a high risk of being captured and compromised. False sensing reports is injected through compromised nodes, which might cause not solely false alarms, however additionally the depletion of restricted energy resource during a battery power-driven network. False data injection types of attacks are intercept by using the IoT-Blockchain based infrastructure.

3 Experiment and Analysis

3.1 Used Technology

The technologies used to set up the targeted system can be listed as written.

1. **Calvin Framework**: Ericsson Calvin framework [1] is used for convenient and reliable environment to develop and manage runtime IoT devices and networks and abstracts the hardware and network dependency.
2. **Digital Certificate:** Digital certificate scheme widely common security solution for digital identity problem. This leads to generating a trust chain among certificate authority to its certified branches to trusted entities
3. **Smart Contract:** Smart Contract is a self-employed software agent that ensures to be run in certain predetermined actions and provide a digital agreement structure that does not need any intermediate carry out contract procedure.
4. **Ethereum:** Distributed application scheme forge opposed to hindrance, deception, and intervention from a third party and it is also Turing complete language runs over Blockchain provide the environment for developers who build up and publish distributed applications.
5. **Solidity:** Solidity is an object-oriented programming language that uses for smart contracts development.
6. **Note:** Remix Ethereum IDE [7], web3 library and GETH (go implementation ethereum) [6] are used for smart contract side simulation, development, and deployment.

3.2 Simulation Setup

In this simulation, there are 3 nodes that represent the initialization smart contract chain in the boot node, collector node that first rule control phase and storage node for storing data. IoT sensor devices placed under collector since its data only pass through the collector.

Environment for Security: The proposed system relies on tested and vastly used security schemes for low-level network layers, provides protection solutions against. These types of attacks require inspection on data structure, the identity of data, and data flow control.

Data structure inspection will be controlled by distributed smart contract based rules. Thus provide convenient and immanent reaction against structural attacks. Identity of data is ensured by the PKI scheme. Data flow control provisioned with the HTTPs scheme.

Environment setup for Performance: Since every component of the test environment run fully virtual on one device we can estimate performance on the real devices from the benchmark result of current IoT devices hardware capabilities. [4,5]

In perspective of IoT devices additional workload is limited to making HTTPs request to the collector. Collector wise performance split into two process lines. First, receiving requests from IoT devices then processing. (Identification of the device, rule control based on the identification of requester) Second data hash is signed by a patient smart identification card built in a PKI scheme [2]. Collectors are highly capable of this type of processing. Cloud storage stance to data processing is dependent on the performance scalability of cloud server hardware. The test environment runs on the device has i7 4700MQ Intel processor, 16Gb RAM, and 256Gb SSD hard drive.

Simulation setup steps:

First installation of proper version for mentioned frameworks. After that initializaed of Ethereum private network that uses Proof of Authority as a consensus for local machine using puppet side executable program that comes with the go-ethereum package [6].

Two sealer node added for Proof of Authority consensus model. The rule Manager smart contract is deployed to this network using Remix IDE [7] and web3 node js library.The deployed contracts are shown on Remix IDE [13].Lastly with configuration and deploying mock IoT device into Calvin runtime using Calvin GUI (Figure 3) the simulation setup was finished.

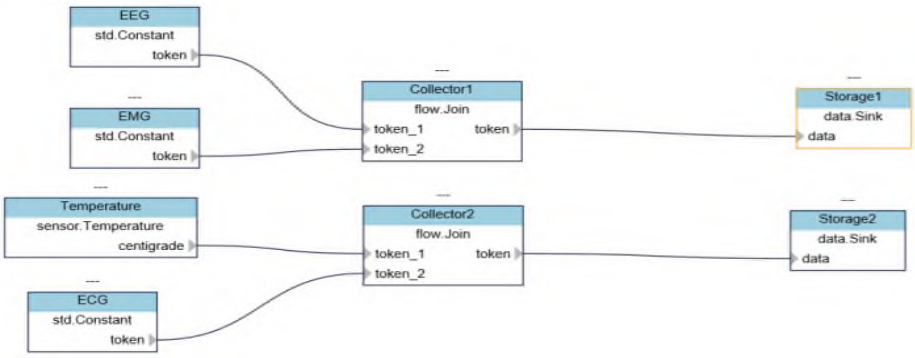


Fig. 3. Deployed mock IoT device into Calvin runtime on Calvin GUI.

3.3 Analysis

In this stage some scenarios were tested and resulted in simulation environment.

Performed Tests.

1. **Load Test on Collector:** In this test we want to measure response time for various counts of requests in the same collector at once. Approximate response time for data

store requests from IoT nodes to collector was measured. In the network configuration block mining time was set to 5 seconds. Thus leads to minimum response time to 5 seconds. Moreover, simulated collector handled specific count of requests. As shown from Figure 5 performed 10, 100 and 1000 simultaneous requests resolved in 5.72, 10.81 and 30,25 seconds consecutively.

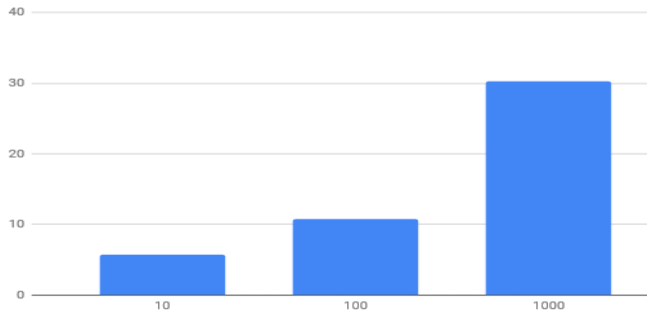


Fig. 4. Request / Response count

2. **Secure Communication Test:** We used openSSL [8] to generate certificates for authority and X509 certificates for IoT nodes, and collectors. Afterwards, certificate authority signed generated certificates for IoT devices and collector nodes. During the tests, simulated collector verified sender IoT nodes' certificates with challenge against certificate authority trust chain control. We observed that If the sender's supplied certificate was not in trust chain of certificate authority this request was rejected by simulated collector as expected. The simuşated nodes deny unsecure HTTP request by default.
3. **Using certificate extension field to interact rule smart contract:** In this test, we checked whether our system successfully extracts requester IoT node group identifier from supplied certificate X509 extension field. Thus leads to acquire trusted information about the identified sender with non repudiation property. Then performed smart contract tests shows that proper smart contract transaction committed based on group identifier.
4. **False data injection attack control:** To test against this type of attack, one of the simulated IoT nodes acted as compromised and sent false data to the simulated collector. Subsequently, the collector verified compromised node certificate. Then, commited new rule smart contract transaction with supplied group identifier in certificate. Therefore, the result of transaction was negative. Thus, proposed system prevents false data to transmitted cloud-storage.
5. **Ethereum Private Network Initialization:** In system preparation phase, we conducted a test deployed collector and cloud-storage nodes properly bootstrapped their blockchain miners. For that one bootnode was deployed to test environment act as negotiator. Simulated collector and cloud-storage were used as sealer nodes. Afterwards, we identify two sealer nodes start to negotiate block mine phase synchronously in nodes' transaction logs.

6. **Deploying Rule Smart Contract:** To test this scenario we generated ethereum private account as authorized system user and deployed it as private network with one bootnode, one sealar, one transaction node. Then deploy smart contract with authorized account to ethereum private network using geth command line tools. Subsequently, we observed that deployed smart contract acknowledged with nodes in private network.

4 Related Works

How to implement IoT in secure way and what are intended solutions' positive and negative sides were investigated in many different prior works. Some of the related studies are explained and compared to our study.

In [9], they suggested a framework called "Privacy Protector", which aims to give protection for patient's privacy, and protected data collection in IoT based healthcare applications because there are attacks against patients' data such as collusion attacks and data leakage and destruction attacks. The framework has two main properties: (i) the secret sharing uses Slepian Wolf coding based secret sharing (SW-SSS) in an attempt to optimize the secret share size, (ii) the other one is share repairing if the data loss or compromise. To apply these properties, they collect data from IoT devices to a communication service provider (CSP). CSP uses the SW-SSS scheme to forward data. In our approach, we use the collectors instead of CSPs for collecting patients' data. In the case of loss or compromise of the data, CSP creates and shares the data again. Doing lots of calculations are required, when we think about lots of the devices sending the data to the collector. To avoid additional computation to repair data, we define rules for the device groups, and according to the rules we control the behavior of the devices. In this way, we prevent data loss or corrupted data.

And the other focus area of mentioned project is while storing data, the mentioned project provides data storage security with using data access control method, in our project all user nodes have public private key pairs, digital certificates and accessing and transmitting data.With this aspects while mentioned project is focusing on only access control and data protection on storage, in our project has multiple security prevention solutions both while collection, transmission , accessing , storing steps multilevel.

The article [10] mentions about cyber threats to personal medical devices and they present solution approaches. They suggested that used to lightweight cryptographic protocols for encryption of patient's data, then record on the cloud-based PHR (Personal Health Record) service. The data can be seen by authorized entities and verification of these entities can be done by using symmetric or asymmetric cryptographic protocols.

In this study, they work with personal medical devices, so devices are used by a specific person. However, in the hospital environment, devices can be assigned, different people. To deal with this situation, we match the patient and the device at the application level by authorized people. In the article, the encryption scheme is designed for medical devices. However, the constraint on the medical devices' memory causes the

limited encryption scheme. This is a disadvantage for the system because it is vulnerable to cyber-attacks. Due to these memory limitations, we offer a rule-based approach by building up a smart contract scheme and use collectors, which have higher capacities than IoT devices.

In the paper [11], they suggest a secure and lightweight authentication scheme using elliptic curve cryptography for IoT based e-health applications. They collect data from nodes to a head node, then, sends it to the base station. A head node is an IoT device with a limited capacity to make calculations. In this way, they studied with the small size of the keys.In our approach; we studied with collectors to prevent limited memory issues. Collectors take the hash of the data with the patients' private keys to protect the integrity of the messages and to provide authentication, we also suggest the certification scheme.

Table 1. Comparison of related works against our project for attack types.

Project/Paper	Attacks					
	Eavesdroppin	Collusion	False data inj.	Sinkhole attack	Reply attacks	Others
PrivacyProtector [9], 2018	prevented	weak	prevented	weak	weak	
Cyber Security for Personal Medical Devices IoT [10], 2014	prevented	prevented	weak	weak	weak	
Secure and Lightweight Auth. Scheme for IoT based E-health Applications [11], 2018	weak	weak	prevented	prevented	weak	Man in the middle
IoTChain: [12], 2018	prevented	weak	weak	weak	weak	
Our Proposed Solution	prevented	prevented	prevented	prevented	prevented	spoofing, node capture

An Ethereum Blockchain based approach to provide security to the IoT devices is used at Alphand et al. [12]. They offer a security architecture called IoT Chain to provide access control to the IoT devices and use the OSCAR scheme and the ACE framework to secure authorization while accessing the IoT resources. IoT chain use Blockchain generate access control tokens with Ethereum smart contracts scheme. In our system smart contracts use for rule policy control for transmitting data.

In our study and their studies have similar approachs like: Both smart contracts are used and combined with the Blockchain. In their study, a smart contract is defined by the resource owner. In our study, a smart contract is published by an authorized person (admin of the system). For authentication of the clients or IoT devices, both studies are used third parties (i.e key server, certification authority).

5 Conclusion

In this paper, we have investigated the effects of using the IoT-Blockchain based infrastructure in healthcare system. In 3-layer architecture which consist of IoT devices, Collectors and Application Layers all nodes are certificated. With intended approach, smart contracts were used to apply rule based access to nodes in order to check data contents of devices. For all communication between layers, rules in smart contracts are

verified and security and data privacy are thus provided. As in the table 1 and table 2 the proposed solution shows that the rule based smart contract IoT system may be used as a preferable solution to prevent most known attacks to provide secure communication environment and to achieve main security goals. As a result of performed tested in this project analysis shows that IoT - Smart contract rule based schema offers solution for common security problems in IoT sensor networks.

References:

1. EricssonResearch - https://github.com/EricssonResearch/calvin-base/wiki/Security
2. Method for enabling PKI functions in a smart card - https://patents.google.com/patent/US7024226B2/en
3. Advantages of Public Key Technology, Safelayer D - https://www.safelayer.com/en/resources/59-articles/public-key-infrastructure/421-advantages-of-public-key-technology
4. J. Esquiagola, L. Costa, P. Calcina, G. Fedrecheski, M. Zuffo, "Performance Testing of an Internet of Things Platform" , Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security - 2017
5. R. Morabito, "A performance evaluation of container technologies on Internet of Things devices", IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) – 2016
6. Puppeth - https://github.com/puppeth/go-ethereum
7. Remix IDE Documentation. https://remix.readthedocs.io/en/latest. (accessed 05 29, 2019)
8. Openssl https://www.openssl.org/(accessed 05 25)
9. E. Luo, M. A. Bhuiyan, G. Wang, M. Rahman, J. Wu, and M. Atiquzzaman, Privacy Protector: Privacy-Protected Patient Data Collection in IoT-based Healthcare Systems,IEEE Communication Magazine (COMMAG), 56(2): 163-168, 2018
10. A. Mohan, "Cyber security for personal medical devices Internet of Things," Proceedings of the IEEE International Conference on Distributed Computing in Sensor Systems, 26-28 May, 2014, pp. 372-374.
11. M. Almulhim, N. Zaman, "Proposing secure and lightweight authentication scheme for IoT based E-health applications", *Advanced Communication Technology (ICACT) 2018 20th International Conference on*, pp. 481-487, 2018.
12. Olivier Alphand, Michele Amoretti, Timothy Claeys, Simone Dall 'Asta, Andrzej Duda, et al..IoT Chain: Blockchain Security Architecture for the Internet of Things. IEEE Wireless Communications and Networking Conference, Apr 2018, Barcelona, Spain. 2018.

3B Modelleme ve Düzenleme Araçlarının Estetik Cerrahi Alanında Kullanılması

İlkin Sevgi İşler[1], Mehmet Harun Doğanay[1], Mürüvet Gökçen[1] ve Ulaş Güleç[2]

[1] Hacettepe Üniversitesi, Bilgisayar Mühendisliği Bölümü, Ankara, Türkiye
{ilkinsevgi,mehmet.doganay,muruvet.gokcen}@hacettepe.edu.tr
[2] Hasan Kalyoncu Üniversitesi, Bilgisayar Mühendisliği Bölümü, Gaziantep, Türkiye
ulas.gulec@hku.edu.tr

Özet. Günümüzde estetik cerrahi, bireyler tarafından oldukça talep gören bir alan olup, bu alanda yapılan ameliyat sayısı gün geçtikçe hızlı bir şekilde artmaktadır. Çoğunlukla bir rahatsızlık nedeni olmadan sadece güzelleşme amacıyla yapılan bu ameliyatlarda, hem hasta beklentilerinin gerçek olamayacak kadar yüksek olmasından hem de doktorların hasta beklentilerini tam olarak anlayamamasından ötürü hastalar bekledikleri görüntüyü elde edememektedir. Tamamen daha güzel bir görünüme sahip olmak için ameliyat olma riskini göze alan hastalar, ameliyat sonrasında bekledikleri görünümü elde edemediklerinde psikolojik açıdan olumsuz bir şekilde etkilenmektedir. Bu nedenle bu çalışmada, ameliyat sonrasında oluşabilecek istenmeyen durumları en aza indirgemek amacıyla doktorların burun ameliyatları öncesinde hastaların beklentilerini görselleştirebileceği bir araç geliştirilmiştir. Hastanın biri önden diğeri ise profilden olmak üzere toplam 2 adet 2B fotoğrafından derin öğrenme tabanlı bir algoritma ile 3B yüz modelini oluşturan bu araçta doktorlar, hastaların 3B modelleri üzerinde çeşitli farklı düzenlemeler yapabilmektedir. Geliştirilen uygulamanın kullanılabilirliğini ölçmek amacıyla uygulama, alanında uzman 8 doktor ile test edilmiştir. Testlerden elde edilen sonuçlara göre geliştirilen bu sistemin, burun ameliyatları öncesinde hem hastaların bekledikleri görünümü görselleştirmede hem de doktorların ameliyat sırasında yapabileceği görünümü oluşturmada kullanılabilecek faydalı bir araç olduğu görülmüştür.

Anahtar Kelimeler: Estetik Cerrahi, Rinoplasti, 3B Modelleme, 3B Görselleştirme, 3B Model Düzenleme

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Usage of 3D Modelling and Morphing Tools in Plastic Surgery

Ilkin Sevgi Isler[1], Mehmet Harun Doganay[1], Muruvet Gokcen[1] and Ulas Gulec[2]

[1] Hacettepe University, Department of Computer Engineering, Ankara, Turkey
{ilkinsevgi,mehmet.doganay,muruvet.gokcen}@hacettepe.edu.tr
[2] Hasan Kalyoncu University, Department of Computer Engineering, Gaziantep, Turkey
ulas.gulec@hku.edu.tr

Abstract. Nowadays, aesthetic surgery is an area that is highly demanded by individuals. In addition, the number of operations performed in this area is also increasing rapidly. In these surgeries performed mostly for the purpose of beautification without any health problems, patients cannot obtain the image they expect since both the patient expectations are too high and the doctors cannot fully understand the patient expectations. Patients who get the risk of having surgery are negatively affected when they cannot obtain the expected appearance after surgery. Therefore, in this study, a tool has been developed in which doctors can visualize patients' expectations before nose surgeries in order to minimize the undesired postoperative conditions. In this tool, which creates a 3D face model with a deep learning based algorithm from a total of 2 2D photographs, one from the front and the other from the profile, doctors can make various arrangements on the 3D models of the patients. In order to measure the usability of the developed application, it was tested with 8 doctors specialized in the field of aesthetic surgery. According to the results obtained from the test, this tool can be efficiently used in order to visualize the patients' expected appearance before the nose operations and what the doctors can perform during the surgeries.

Keywords: Plastic Surgery, Rhinoplasty, 3D Modelling, 3D Visualization, 3D Model Morphing

1 Giriş

Günümüzde, sosyal medya ve pazarlamanın bireyler üzerinde yarattığı etki ile birlikte birçok kişide filmlerde ve dizilerde parlatılan 'kusursuz' insanlara benzeme isteği oluşmaktadır. Bu durum sonucunda çoğu insan, kendini yetersiz hissederek güzelleşme uğruna tıbbi prosedürlere başvurmaktadır [7]. Birçok farklı alanda uygulanabilen bu tıbbi prosedürlere bakıldığında ise burun ameliyatı, burnun üstlenmiş olduğu kritik fonksiyonel işlevlerinin yanı sıra kişilerin dış görünüşlerine karakteristik özellikler katmasından dolayı en çok tercih edilen ameliyat türlerinden biri olarak ortaya çıkmaktadır [6]. Her ne kadar burun ameliyatları

popüler bir ameliyat türü olsa da, ameliyat olma kararı birçok kişi için çok zorlu bir süreçtir [14]. Özellikle, sağlık sorunu olmayan bireylerin tercihen oldukları bu ameliyatlarda beklenen sonuçları alamama ihtimali, kişilerde endişe ve kaygı yaratmaktadır. Üstelik bu ihtimal, prosedür sonucu burun yapısının kendilerinde aynı şekilde görünüp görünmeyeceğini bilememesinden dolayı oldukça yüksektir [20]. Bu tip istenmeyen durumları ortadan kaldırarak bu ameliyatlardaki başarı oranını artırmak için, kişiye yapılacak burun modelinin kişilerin kendi yüz hatlarıyla bir uyum ve bütünlük içinde olması gerekmektedir. Bu bütünlüğe de hastayla birlikte prosedür öncesi karar vermek ve hastanın da fikir ve beklentileri doğrultusunda hareket etmek prosedür sonrası iki tarafı da mutlu etmektedir. Bu sayede hasta da yüksek beklentilere girmeden burnun fonksiyonel işlevinin en üst düzeyde korunduğu aynı zamanda da görüntü olarak istenen bir burna kavuşabilmektedir. Bu kapsam doğrultusunda belirtilen problemi çözmek adına literatürde kişilerin 2 boyutlu resimleri üzerinde değişiklik yapmalarını sağlayan birçok farklı çalışma bulunmaktadır [19,21]. Ancak, ameliyat olma kararı aşamasında çekilen hasta fotoğraflarının 2 boyutlu halleri üzerinde değişiklikler yapmak çoğu zaman hastanın beklentilerinin anlaşılmasında yetersiz kalmaktadır [18]. Bu yetersizliğin oluşmasındaki en büyük sebep ise fotoğraf üzerinde çekilen andaki ışığın tam olarak gerçeği yansıtamamasıdır [24]. Düzenleme sırasında ışık da dokularla hareket ettiği icin gerçekçi bir görüntü elde edilememektedir. Buna ek olarak, hastanın burnunu her açıdan görememesi de var olan uygulamaların önemli bir eksikliğidir [3].

Literatürde var olan çalışmaların belirttigi eksiklikleri kapatmak icin geliştirilen bu projede, hastanın önden ve yandan çekilmiş fotoğraflarıyla üzerinde düzenlemelerin gerçekleştirilebileceği 3 boyutlu bir model oluşturulmaktadır. Bu sayede doktorların 3B model üzerinde yapacağı düzenlemeler ile daha gercekçi sonuçlar elde edilerek bu sonuçların hastalara kolay bir sekilde anlatılması amaçlanmıştır. Bildirinin geri kalan kısmı şu şekilde oluşturulmuştur: Bir sonraki bölümde çalışmanın alt yapısını oluşturmak için literaturdeki benzer çalışmalar detaylı bir şekilde anlatılmıştır. Üçüncü bölümde, sistemin tasarımı açıklanarak kişilerin sistem ile ilgili düşünceleri paylaşılmıştır. Dördüncü bölümde ise elde edilen sonuçlar tartışılarak bildiri sonlandırılacaktır.

2 Literatür Değerlendirmesi

Estetik ameliyatlar, görünüşlerinden memnun olmayan kişilerin kendilerini daha iyi hissetmek adına yaptırdıkları operasyonlardır [27]. Genellikle zorunlu olmayan bu prosedürlerin sağlıksal bir sebep yerine güzellik kaygısından yapılıyor olması hastanın beklentisini arttırdığından, ameliyat sonrasındaki sonucu bilememek hastanın tedirginliğini artırmaktadır. Sürecin başındaki zorlu karar aşamasından dolayı hastaya beklenilen sonucun görsel olarak gösterilmesi hasta-doktor iletişimini kolaylaştırmakta ve hastanın kendinden emin bir şekilde karar vermesine yardımcı olmaktadır [15]. Bu sebeple, hem hasta memnuniyetini artırmak hem de doktorların işini kolaylaştırmak adına, literatürde 2B ve 3B modelleme/düzenleme araçları geliştiren birçok çalışma bulunmaktadır. Bu kapsamda

2B modeller üzerinde yapılan çalışmalar incelendiğinde, Chabanas ve diğerlerinin [5] yapmış olduğu çalışmada, ortognatik cerrahi işlemler olan mandibula (alt çene) ve maksilla (üst çene) deformitelerini mesh-matching metodu ile otomatik olarak modelleyen bir uygulama geliştirilmiştir. Geliştirilen uygulamanın başarısını test etmek amacıyla 6 hasta için model oluşturulmuş ve bu modellerden 2 tanesinin simülasyonu yapılmıştır. Simülasyonlardan elde edilen sonuçlar 5 cerrah tarafından değerlendirilmiş olup, geliştirilmiş uygulamanın hastaların çene deformasyonlarını otomatik olarak tespit etmede başarılı bir araç olduğu belirtilmiştir. Rabi ve diğerleri [21] tarafından yapılan diğer bir çalışmada ise kişinin yüzüne, başka bir kişinin yüz hatları ve ten rengi gibi özellikleri otomatik olarak yerleştirilerek hastanın planlanan görüntüyü görmesi hedeflenmiştir. Geliştirilen bu uygulama, hastanın sahip olduğu dokular arasında hiçbir kesinti ve renk farklılığı yaratmadığı için, estetik çene ameliyatı gibi birçok farklı cerrahi prosedür öncesi etkili bir şekilde kullanılması planlanmıştır. Bu kapsamda yapılan diğer bir çalışmada ise burun estetik ameliyati öncesinde hastaya burnun anatomik yapısı göz önünde bulundurularak oluşan görüntüyü sunan bir uygulama geliştirilmiştir [19]. Diğer çalışmalardan farklı olarak bu çalışmada burun, belirli kısıtlamalar çerçevesinde düzenlenirken, ideal bir burnun sahip olduğu yapısal formun korunmasına yönelik yönlendirmeler bulunmaktadır.

Yukarıdaki çalışmalardan da görüleceği üzere literatürde 2B görüntüleme üzerine birçok çalışma mevcuttur. Ancak hasta yüzünün görselleştirilmesi ile ilgili 2B görüntüleme teknikleri kullanıldığında bu teknikler ile üretilen görüntülerin ışıklandırmalarının yetersizliği [24] ve yüzün her açıdan görünememesi [18] gibi sebeplerden ötürü hastaların beklentilerini karşılayamayarak yetersiz kaldığı görülmüştür. Tam bu noktada, 3B modeller belirtilen bu eksiklikleri ortadan kaldırarak daha gerçekçi görüntülerin elde edilmesinde kullanılmaktadır.

Literatürde 3B modellerin oluşturulmasıda birçok farklı yöntem bulunmaktadır. Bu yöntemlerden ilki bilgisayarlı tomografi yöntemidir [12,13]. Bu yöntemle istenilen sonuca ulaşılsa da, maruz kalınan radyasyon ve maliyet göz önüne alındığında estetik prosedüre uygun olmadıkları anlaşılmıştır. Diğer bir yöntem olan lazer ve fotogrometrik tarayıcılar ile istenilen yüz şekillerinin 3B modelleri de etkili bir şekilde oluşturulabilmektedir [9,16,23]. Ancak bu yöntemler ile modellerin oluşturulması işlemi çok maliyetli, donanım bağımlı ve kullanımı karmaşık olduğu için kişiler tarafından erişilmesi zordur. Wang ve diğerleri [26] tarafından yapılan diğer bir çalışmada ise 3B lazer tarayıcı ve lateral X-ray fotoğrafı kullanılarak yüksek hassaslıkta modeller oluşturulmaktadir. Bu yöntemde derinin kalınlığı X-ray fotoğrafından girdi olarak alınarak mode manuel veya istatiksel verilere göre otomatik olarak modifiye edilebilmektedir. Sistemin başarısını anlayabilmek için geliştirilen uygulama bu alanda çalışan cerrahlar tarafından test edilmiştir. Bu testlerden elde edilen sonuçlara göre sistemin, günlük klinik çalışmalara yardımcı olabileceği belirtilmiştir. Her ne kadar başarılı modeller elde edilse de bu modelleri oluşturmak için kullanılacak lazer tarayıcı ve x-ray cihazının maliyetli olması bu yöntemin kullanılmasını zorlaştırmaktadır. Sonuç olarak, bahsedilen araçların teknik ve finansal açıdan kullanımının zor olması bu araçların kullanımının yaygınlaşmasına engel olmaktadır.

Yukarıda belirtilen yöntemlere alternatif olarak, fotoğraf ve video kullanılarak ton farkı [28], hareket [10, 25] ve gölgelendirme [17, 22] yöntemlerinden yararlanılarak donanımdan bağımsız oluşturulan 3B modeller de bulunmaktadır. Oliveira-Santos ve diğerleri [18] tarafından tamamlanan çalışmada burun estetik ameliyatı öncesinde hastaya oluşacak görüntüyü sunan web tabanlı bir uygulama geliştirilmiştir. Bu çalışmadan elde edilen sonuçlara göre, geliştirilen sistem burun ameliyatlarında kullanılabilecek faydalı bir araçtır. Baumberger ve diğerleri [2] tarafından yapılan diğer bir çalışmada ise hastanın video görüntüsünden 3B modeller oluşturulmaktadır. *Spatio-temporal* takip etme metodu ile oluşturulan modeller %5 medyan hata payına sahiptir. Video takip metodu sayesinde oluşturulan modellerin başarısında kullanılan fotoğraf sayısının önemli olmadığı anlaşılmıştır. Amberg ve diğerleri [1] tarafından yapılan çalışmada ICP algoritması kullanılarak duygusal değişimlerden bağımsız olarak 3D model oluşturulmuştur. Çekilen fotoğraflarda kişi gülüyor veya kızıyorsa bile gerçekçi bir 3B model oluşturulmaktadır. Elde edilen yüz modellerindeki doğruluk oranı doğal modelle 98.1% iken duyguların olduğu modellerle 99.7%'e kadar çıkmaktadır.

Özet olarak, literatürde estetik ameliyatları öncesinde hastaların kaygı seviyelerini düşürerek memnuniyetlerini artırmak amacıyla hem 2B hem de 3B modeller kullanan birçok bilgisayar tabanlı uygulama bulunmaktadır. Ancak, hem 2B modellerin gerçeği tam olarak yansıtamaması hem de 3B model kullanan uygulamaların çok maliyetli olmasından dolayı bu uygulamalar istenilen seviyede kullanılamamaktadır. Bu doğrultuda, bu çalışmada açık kaynaklı araçların modifiye edilerek 2B fotoğraflardan 3B modellerin etkili bir şekilde oluşturulması ve oluşturulan bu modellerin üzerinde çeşitli birçok farklı düzenleme yapılması hedeflenmiştir. Bir sonraki bölüm, geliştirilen geliştirilen bu uygulamayı detaylı bir şekilde açıklayacaktır.

3 Yöntem

3.1 Kullanılan Araçlar

Bu çalışmada, estetik ameliyatların öncesinde doktorların hastaları ile iletişimini kolaylaştırmak ve ameliyat sonrası beklenen görüntüyü hastaya anlatmaya yardımcı olmak amacıyla hastaların fotoğraflarından 3B model oluşturan bir araç geliştirilmiştir. Bu araçta, oluşturulan model üzerinde düzenlemeler yapabilmek için Python programlama dili ile Blender 3D programı ile uyumlu bir şekilde çalışan bir arayüz tasarlanmıştır.

Bu teknolojilerden Blender 3D; modelleme, animasyon, hareket izleme ve oluşturma gibi işlemleri tamamen destekleyen açık kaynaklı ve ücretsiz bir 3B modelleme ortamıdır [4]. Geliştirilen arayüz ile Blender 3D'de doktorların hastanın fotoğraflarını seçebilecekleri ve seçilen fotoğraflardan oluşturulan 3B model üzerinde düzenlemeler yapabilecekleri menüler arayüze entegre edilmiştir.

Python programlama dili, gerek makine öğrenmesi ve derin öğrenme çalışmalarında hızlı çalışıyor olması, gerekse sahip olduğu geniş standart kütüphaneleri ile oldukça popülerleşmiş ve modelleme alanında geniş kullanım alanına sahip

bir programlama dilidir [11]. Python programlama dili, Blender 3D programı ile entegre bir şekilde çalıştığı için bu çalışmada tercih edilmiştir.

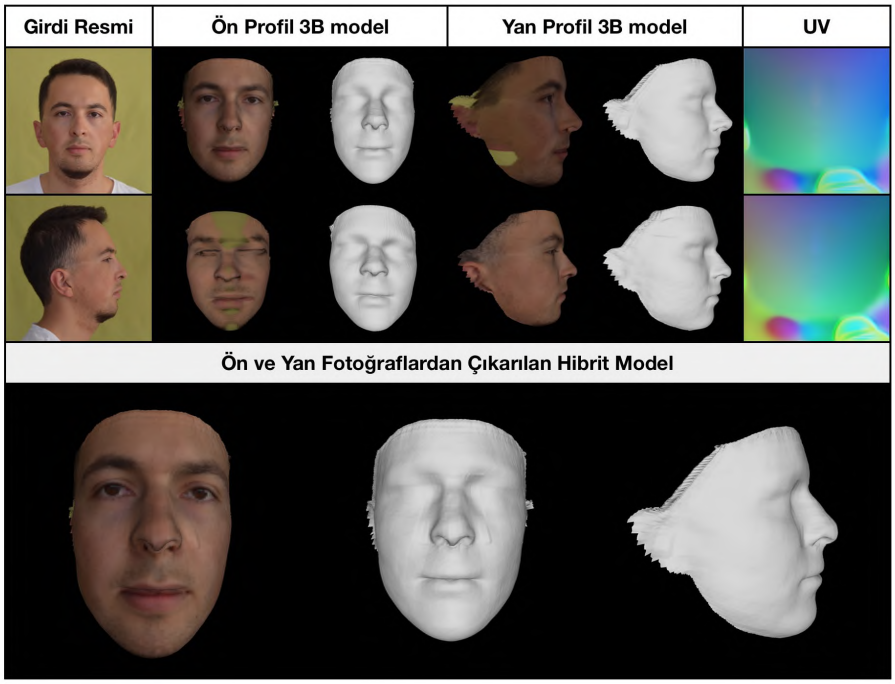
3.2 Sistem Tasarımı

Bu proje, 3B yüz modelinin oluşturulması ve oluşturulan 3B modelin düzenlenmesi olarak iki ana bölümden oluşmaktadır. Amaç, 3B yüz geometrisini ve insan yüzünün doku bilgilerini 2B fotoğraflardan elde etmektir. Bu amaç doğrultusunda geliştirilen sistemde, biri profil fotoğrafı diğeri ise önden bir fotoğraf olmak üzere toplam 2 adet fotoğraf ile 3B yüz yapısını eşzamanlı olarak yapılandıran ve yoğun hizalama sağlayan bir model oluşturulmaktadır. Belirtilen bu işlevleri sağlayabilmek için bu çalışmada, yüzün 3B şeklini UV uzayına kaydeden UV konum haritası adı verilen bir 2B gösterim tasarlanarak, bu 2B görüntüden 3B modeli oluşturmak için Evrişimsel Sinir Ağlarını eğiten bir algoritmadan faydalanılmıştır [8]. Bu sayede, burun ameliyatlarında faydalı olacak, gerçekçi ve beklenen görüntüyü hastaya anlatmaya yardımcı olacak bir sistem geliştirmek hedeflenmiştir.

Çalışmanın amacı 3B modellerin burun estetiği alanında kullanılması olduğu için, hastaların burun bölgelerinin daha detaylı ve gerçeğe yakın olması gerekmektedir. Bu amaçla yüzdeki belirli kısımların profilden çekilmiş fotoğraflardan elde edilen model bilgisinin kullanılması gerekmektedir. Bu sebeple hastaların 3B yüz modelleri oluşturulurken, burun ve çene gibi insan yüzündeki önemli olan bölgelerde, hastanın yan profil fotoğrafından kesilen bölgeden oluşturulan UV konum haritası ile ön fotoğrafından oluşturulan UV konum haritasını birleştirilerek yüz modeli oluşturulmaktır.

Amaç 3B yüz geometrisini ve yoğun hizalama bilgilerini 2B fotoğraflardan elde etmektir. Bu sebeple derin ağların doğrudan 2B fotoğraflardan elde edilen bilgiler üzerinden modelleme yapabilmesini sağlayacak bir gösterim kullanılmıştır. Şekil 1'de gösterildiği üzere UV konum haritası, 3B tüm yüz yapısını ve hizalama bilgisini de içeren haritayı sunmaktadır. Bu harita, UV uzayındaki tüm noktaların 3B konumlarının kayıtlı olduğu 2B bir görüntüdür. Derin ağ algoritması girdi olarak RGB görüntüyü konum haritasına çevireceği için, çevirme işlemini öğrenmek adına bir *encoder-decoder* yapısı kullanılmıştır. Fotoğraflara karşılık gelen konum haritalarının bulunması için 3DMM parametreleri fotoğrafları UV uzayına dönüştürmek için kullanılmıştır. Yüzün merkezi diğer bölgelere göre daha ayırt edici özelliklere sahip olduğundan ağı eğitmek için bir ağırlık maskesinden faydalanılmıştır. Amaç yüzü net bir şekilde modellemek olduğu için, yüzdeki noktalar 4 parçaya ayrılmıştır ve bu dört parçanın ağırlıkları birbirinden farklı olarak belirlenmiştir. Ağı eğitmek için farklı açılardan çekilmiş birden çok fotoğrafın bulunduğu ve bu fotoğraflardan 3B nokta bulutunun kolayca oluşturulabileceği 300W-LP [29] veri seti kullanılmıştır. Eğitim setindeki fotoğraflar rastgele farklı açılarda döndürülerek ve çevirilerek tüm zor durumları ele almak için çeşitlendirilmiştir.

Geliştirilen program, Blender programı üzerinde entegre bir şekilde çalışmaktadır. Blender programı için geliştirilen arayüzler, doktorların kolay bir şekilde kullanabilecekleri bir biçimde tasarlanmıştır. İlk olarak doktorun, hastanın 3B



Şekil 1. 3B Modelin Çıkarılması

yüz modelini oluşturabilmesi için hastanın fotoğraflarını sisteme yüklemesi gerekmektedir. Bu işlemi yapabilmek için doktorun, Şekil 2'de gösterilen menüdeki "*Yüz Modeli -> Fotoğraf Seç*" sekmesine tıklayarak hastanın fotoğraflarının olduğu klasörü seçmesi gerekmektedir. Fotoğraf seçimi Şekil 3'te gösterildiği gibi yapılmaktadır. Doktor hastanın fotoğraflarının olduğu klasörü seçtikten sonra *Dosya Yolunu Al* tuşuna tıklamalıdır. Fotoğrafları seçilen hastanın 3B yüz modeli arka planda oluşturulur. Ardından Blender'ın "*3D View*" ekranında gözükür.

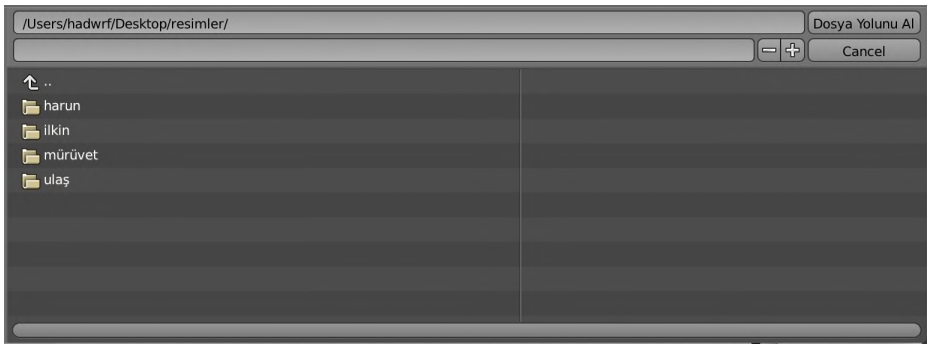
Yüz Modeli menüsünde *Görüntüleme Modu* ve *Düzenleme Modu* olmak üzere 2 tane mod vardır. Hastanın 3B yüz modeli üzerinde yapısal değişikliklerin yapılabilceği mod düzenleme modudur.

Düzenleme modu seçildiği zaman ekranın sol tarafında fırçaların görüldüğü bir araç çubuğu ekrana gelir. Bu kısımda Şekil 2'de görülen çeşitli fırçalar seçilebilmektedir. Bu fırçalarla insan yüzü üzerinde yapılmak istenen yapısal değişikliklerin yapılmasına olanak sağlamaktadır.

Doktor, 3B modele farklı açılardan bakmak isteyebilir. Bunu imleç yardımıyla da yapabilir. Fakat önden, sağ-sol profilden, aşağıdan ve yukarıdan da bakmak istenebilir. Bunun gibi açılarda 3B modeli döndürebilmek için menü çubuğunda



Şekil 2. Blender Arayüzü ve Menüler



Şekil 3. Hasta Resimlerinin Uygulamaya Yüklenmesi

Görüntü Açısı adı altında bir menü oluşturularak alt menülerde bu açılardan görüntüyü ekrana yansıtacak seçenekler eklenmiştir. Aynı zamanda doktor belirli bir bölgede daha detaylı çalışma yapmak isteyebilir. Çalışma yapacağı alanı daha kolay seçebilmesi için *Seçili Bölgeye Yaklaş* seçeneği konulmuştur. Bu seçeneğe tıkladığımızda ekranda dikdörtgensel alanı seçmemize olanak sağlayan bir araç belirir. İmleci basılı tutarak istenilen alan seçildiğinde o bölgeye yaklaşılır.

Menü çubuğundaki menülerden birisi ise *Maskeleme* menüsüdür. Maskeleme aracı, doktorun sadece kendi belirlediği alanda düzenleme yapmasını sağlayacak bir araçtır. Diktörtgen ve Lasso maskeleme araçları mevcuttur. Lasso maskeleme

aracı SHIFT+CTRL ile aktive edilip istenilen bölge maskelenebilir. Lasso maskeleme aracı dikdörtgensel olmayan bölgeleri seçmede daha iyi sonuçlar verirken dikdörtgen maskeleme de alternatif olarak kullanılabilir. Maskeleme araçları ile belirli bir bölgeyi seçtikten sonra maskelenmiş alanın zıttı da *Maskelemeyi Tersine Çevir* seçeneği ile aktive edilebilr.

3B model üzerinde değişiklikler yapılırken, doku olmadan da çalışılabilir. Bunun için ALT+Z tuş kombinasyonu ile 3B model üzerindeki doku aktive ve deaktive edilebilir. Dokunun olmadığı gösterim (bkz. Şekil 4) 3B model üzerinde daha hassas çalışılabilmesine olanak sağlar.



Şekil 4. 3B Model Üzerinde Yapılan Değişiklikler. 1.satır girdi modeli, 2. satır düzenlenmiş model

Yapılan çeşitli düzenlemeler sonucunda Şekil 4'teki gibi bir sonuç elde edilmiştir. 1. satırdaki görüntüler programa girdi olarak verilen resimlerden elde edilen üzerinde oynama yapılmamış 3B modellerdir. 1. sütun profilden gösterim, 2. sütun önden gösterim, 3. sütun ise dokunun olmadığı gösterimdir. 2. satır ise doktorun üzerinde çeşitli düzenlemeler yaptığı modelin görüntüleridir. Bu görüntüyü elde etmek için, *Thumb, Inflate/Deflate, Crease* ve *Smooth* fırçaları kullanılmıştır.

4 Deneyler ve Analizler

Geliştirilen bu sistem, plastik cerrahi ve kulak-burun-boğaz alanında 10 yılı aşkın tecrübesi olan 8 uzman doktor ile değerlendirilmiş olup, doktorların sistem ile ilgili düşüncelerini alabilmek için aşağıdaki sorular sorulmuştur.

S1: Burun ameliyatları icin 3B modelleme yapan programların varlıgından haberdarım.
S2: Hasta, operasyon sonrasında ortaya cıkacak goruntuyu gormek ister.
S3: 3B model gormek hastayı ameliyata ikna etmeyi saglar.
S4: Operasyon oncesi simulasyonda bize yardımcı olur.
S5: Modelin gercekci bir sonuc verdigini dusunuyorum.
S6: Doktor acısından kullanması kolay bir programdır.
S7: Modelleme programları ile ugrasmak doktorlar icin vakit kaybıdır.
S8: Bu urunu klinigimde kullanmak isterim.

Uzmanlar yukarıdaki sorulara cevap verebilmek için her soruya *1: Kesinlikle katılmıyorum*, *2: Katılmıyorum*, *3: Ne katılıyorum ne katılmıyorum*, *4: Katılıyorum* ve *5: Kesinlikle katılıyorum* olacak şekilde Likert ölçü sistemine uygun bir notlandırma sisteminde her soruya puan vermiştir. Uzmanların verdiği notlar Tablo 1'de gösterilmiştir.

Tablo 1. Sistem ile İlgili Uzman Kişilerin Değerlendirme Sonuçları

	S1	S2	S3	S4	S5	S6	S7	S8
1. Uzman	5	4	4	4	4	4	3	3
2. Uzman	5	5	5	5	4	-	2	5
3. Uzman	5	5	5	5	4	-	1	5
4. Uzman	5	5	4	5	3	4	2	3
5. Uzman	5	4	4	4	4	4	3	3
6. Uzman	4	3	4	4	3	4	3	-
7. Uzman	5	5	4	4	2	5	3	-
8. Uzman	2	5	4	4	4	3	2	4
Ortalama	4.5	4.5	4.3	4.4	3.3	4	2.4	3.8

Uzmanların verdiği cevaplar soru bazında ortalama skor alınarak incelendiği zaman, bütün sorularda (7. soru ters soru olduğu için 2.4 başarılı bir skor olarak değerlendirilmiştir.) uzmanların sistem ile ilgili verdikleri görüşlerin olumlu olduğu görülmüştür. Doktorların birçoğu, böyle bir sistemin ameliyat öncesinde hem hastanın beklentilerinin anlaşılmasında hem de hastanın gözünde yapılması planlanan burun yapısının anlaşılarak hastanın endişe seviyesini düşürmede kullanılabilecek faydalı bir araç olduğunu düşünmektedir.

5 Sonuçlar ve Gelecek Çalışmalar

Günümüzde estetik kaygılar sebebi ile ameliyat olan kişilerin sayısı oldukça fazladır. Bu çalışma, hastaların ameliyat sonrası karşılaşacakları görüntü konusunda isteklerinin anlaşılması ve doktorlar ile hastaların iletişimlerini artması amacıyla geliştirilmiştir. Bu amaç doğrultusunda hastanın fotoğraflarından, yüzünün 3B modelinin oluşturulduğu ve doktorun üzerinde 3B değişiklikler yapabileceği az maliyetli ve derin öğrenme tabanlı bir uygulama tasarlanmıştır.

Geliştirilen uygulama hakkında, alanında uzman doktorlarların görüşleri alınmıştır. Uzmanlarla yapılan görüşmelerin sonucunda ise gerçek hayatta bu tür uygulamaların gerekli olduğu ve ulaşılmasının zor olduğu ifade edilmiştir. Ameliyat öncesi bu uygulama kullanıldığında hastanın endişe seviyesinin düşebileceği, doktor için faydalı bir araç olduğu düşünülmektedir.

Gelecek çalışmalarda ise, daha gerçekçi bir 3B model oluşturulması, doktorların kullanacağı arayüze daha fazla fonksiyonların eklenmesi ve hasta yüzü üzerinde yapılan değişikliklerin hasta yüzünde gerçek zamanlı olarak görüntülenebilmesi amaçlanmaktadır.

Kaynakça

1. Amberg, B., Knothe, R., Vetter, T.: Expression invariant 3d face recognition with a morphable model. In: 2008 8th IEEE International Conference on Automatic Face & Gesture Recognition. pp. 1–6. IEEE (2008)
2. Baumberger, C., Reyes, M., Constantinescu, M., Olariu, R., de Aguiar, E., Santos, T.O.: 3d face reconstruction from video using 3d morphable model and silhouette. In: 2014 27th SIBGRAPI Conference on Graphics, Patterns and Images. pp. 1–8. IEEE (2014)
3. Biadgligne, Y.: Performance evaluation of photometric normalization (2015)
4. Brito, A.: Blender 3D: Architecture, Buildings, and Scenery: Create photorealistic 3D architectural visualizations of buildings, interiors, and environmental scenery. Packt Publishing Ltd (2008)
5. Chabanas, M., Luboz, V., Payan, Y.: Patient specific finite element model of the face soft tissues for computer-assisted maxillofacial surgery. Medical image analysis 7(2), 131–151 (2003)
6. Daniel, R.K.: The preservation rhinoplasty: a new rhinoplasty revolution (2018)
7. Darisi, T., Thorne, S., Iacobelli, C.: Influences on decision-making for undergoing plastic surgery: a mental models and quantitative assessment. Plastic and Reconstructive Surgery 116(3), 907–916 (2005)
8. Feng, Y., Wu, F., Shao, X., Wang, Y., Zhou, X.: Joint 3d face reconstruction and dense alignment with position map regression network. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 534–551 (2018)
9. Heike, C.L., Upson, K., Stuhaug, E., Weinberg, S.M.: 3d digital stereophotogrammetry: a practical guide to facial image acquisition. Head & face medicine 6(1), 18 (2010)
10. Jebara, T., Azarbayejani, A., Pentland, A.: 3d structure from 2d motion. IEEE Signal processing magazine 16(3), 66–84 (1999)
11. Joshi, P.: Artificial intelligence with python. Packt Publishing Ltd (2017)

12. Kim, H., Jürgens, P., Weber, S., Nolte, L.P., Reyes, M.: A new soft-tissue simulation strategy for cranio-maxillofacial surgery using facial muscle template model. Progress in biophysics and molecular biology 103(2-3), 284–291 (2010)
13. Koch, R.M., Gross, M.H., Carls, F.R., von Büren, D.F., Fankhauser, G., Parish, Y.I.: Simulating facial surgery using finite element models. Technischer Bericht/Eidgenössische Technische Hochschule, Departement Informatik 246 (1996)
14. Leem, S.Y.: The anxious production of beauty: Unruly bodies, surgical anxiety and invisible care. Social studies of science 46(1), 34–55 (2016)
15. Lekakis, G., Claes, P., Hamilton III, G.S., Hellings, P.: Three-dimensional surface imaging and the continuous evolution of preoperative and postoperative assessment in rhinoplasty. Facial Plastic Surgery 32(01), 088–094 (2016)
16. Lin, S.J., Patel, N., O'shaughnessy, K., Fine, N.: A new three-dimensional imaging device in facial aesthetic and reconstructive surgery. Otolaryngology—Head and Neck Surgery 139(2), 313–315 (2008)
17. Moghaddam, B., Lee, J., Pfister, H., Machiraju, R.: Model-based 3d face capture with shape-from-silhouettes. In: 2003 IEEE International SOI Conference. Proceedings (Cat. No. 03CH37443). pp. 20–27. IEEE (2003)
18. Oliveira-Santos, T., Baumberger, C., Constantinescu, M., Olariu, R., Nolte, L.P., Alaraibi, S., Reyes, M.: 3d face reconstruction from 2d pictures: first results of a web-based computer aided system for aesthetic procedures. Annals of biomedical engineering 41(5), 952–966 (2013)
19. Ozkul, T., Ozkul, M.H.: Computer simulation tool for rhinoplasty planning. Computers in Biology and Medicine 34(8), 697–718 (2004)
20. Pearson, D.C., Adamson, P.A.: The ideal nasal profile: rhinoplasty patients vs the general public. Archives of facial plastic surgery 6(4), 257–262 (2004)
21. Rabi, S.A., Aarabi, P.: Face fusion: an automatic method for virtual plastic surgery. In: 2006 9th International Conference on Information Fusion. pp. 1–7. IEEE (2006)
22. Salzmann, M., Pilet, J., Ilic, S., Fua, P.: Surface deformation models for nonrigid 3d shape recovery. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(8), 1481–1487 (2007)
23. See, M.S., Foxton, M., Miedzianowski-Sinclair, N., Roberts, C., Nduka, C.: Stereophotogrammetric measurement of the nasolabial fold in repose: a study of age and posture-related changes. European Journal of Plastic Surgery 29(8), 387–393 (2007)
24. Suliga, P., Wrona, T.: Microlens array calibration method for a light field camera. In: 2018 19th International Carpathian Control Conference (ICCC). pp. 19–22. IEEE (2018)
25. Torresani, L., Hertzmann, A., Bregler, C.: Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. IEEE transactions on pattern analysis and machine intelligence 30(5), 878–892 (2008)
26. Wang, J.x., Liao, S.h., Zhu, X.h., Wang, Y., Ling, C.x., Ding, X., Fang, Y.m., Zhang, X.h.: Real time 3d simulation for nose surgery and automatic individual prosthesis design. Computer methods and programs in biomedicine 104(3), 472–479 (2011)
27. Wilkie, T.F.: A psychological study of patients undergoing cosmetic surgery. Plastic and Reconstructive Surgery 58(2), 257 (1976)
28. Zhang, R., Tsai, P.S., Cryer, J.E., Shah, M.: Shape-from-shading: a survey. IEEE transactions on pattern analysis and machine intelligence 21(8), 690–706 (1999)
29. Zhu, X., Lei, Z., Liu, X., Shi, H., Li, S.Z.: Face alignment across large poses: A 3d solution. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 146–155 (2016)

Yazılım Hataları ile İlişkili Olmayan Olay Kayıtlarının Analizi ve Sınıflandırılması: Bir Vaka Çalışması

Ethem Utku Aktaş[1,2], Mehtap Özaltın Eriş[1], Haluk Altunel[1], and Cemal Yılmaz[2]

[1] Softtech A.Ş., Arge Merkezi, 34947 İstanbul, Türkiye
{utku.aktas, mehtap.ozaltin, haluk.altunel}@softtech.com.tr
[2] Mühendislik ve Doğa Bilimleri Fakültesi, Sabancı Üniversitesi, 34956 İstanbul, Türkiye
cyilmaz@sabanciuniv.edu

Özet. Yazılım olay kayıtları veya yazılım hata raporları müşterilerin veya testçilerin bir yazılım ürününü kullanırken karşılaştıkları sorunları raporladıkları kayıtlardır. Olay kaydı yönetim sistemleri aracılığıyla kullanıcılar kolay ve hızlı bir şekilde kayıt oluşturabilmekte, yazılımcıların da kısa bir süre içerisinde bu kayıtlara çözüm üretmesi beklenmektedir. Bu sistemler sayesinde yazılım ürününün kalitesinin artırılması ve müşteri memnuniyetinin sağlanması hedeflenmektedir. Ancak kullanıcıların kolay bir şekilde kayıt açabiliyor olması, geliştirme talebi, operasyonel talep veya bilgi eksikliği kaynaklı, aslında yazılım hatası olmayan konularda da kayıt açılmasına neden olmakta, bu durumlar da yazılımcıların ilgili kaydı iade etmesi ile sonuçlanmaktadır. Bu tür kayıtlara yazılımcıların zaman harcıyor olması verimliliklerini düşürmekte, aynı zamanda yazılım olay kaydı süreçlerinin yönetimini de zorlaştırmaktadır. Bu çalışmayla, endüstriyel bir vakaya ait olay kaydı adetlerindeki artış eğilimi analiz edilmiş ve kayıtlar kapatılırken seçilen sebep kategorileri incelenerek yazılım hatası olan ve olmayan kayıtlar ayrıştırılmıştır. Sonrasında kayıtların yazılım hatasından kaynaklanıp kaynaklanmadığı tahmin edilmiş ve yazılım hatasından kaynaklanmayan kayıtların % 77 f-skor ile tahminlenebildiği görülmüştür. Bu ilk çalışma sonucunda, kullanıcılara veya yazılımcılara bir olay kaydının yazılım hatasından kaynaklanıp kaynaklanmadığına dair otomatik olarak uyarı verecek bir sistemin mümkün olduğu görülmüştür.

Anahtar Kelimeler: Yazılım Hata Kayıtları, Bilgi Çıkarımı, Veri Analizi, Veri Madenciliği.

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Analysis and Classification of Issue Reports Not Related to Software Bugs: A Case Study

Ethem Utku Aktaş[1,2], Mehtap Özaltın Eriş[1], Haluk Altunel[1], and Cemal Yılmaz[2]

[1] Softtech Inc., Research and Development Center, 34947 Istanbul, Turkey
{utku.aktas, mehtap.ozaltin, haluk.altunel}@softtech.com.tr
[2] Faculty of Engineering and Natural Sciences, Sabanci University, 34956 Istanbul, Turkey
cyilmaz@sabanciuniv.edu

Abstract. Software issue records or bug reports are records where customers or testers report problems encountered while using a software product. Users can create issue reports easily and quickly through bug report management systems, and software developers are expected to solve these records in a limited amount of time. With these systems, it is aimed to increase the quality of the software product and to ensure customer satisfaction. However, since the users can easily create these records, software development requests, operational requests or issues arising from lack of information can also be reported such that there is no software error or failure in the case. In such cases, software developers return the records without any action. The fact that software developers spend time on such records reduces their efficiency and makes it difficult to manage bug repositories. In this study, the increase in the amount of issue records for an industrial case was analysed and by categorizing the issue reports as bugs or non-bugs according to the reason option selected during the closure of the report, a classification study is conducted. The results show that 77 % f-score can be reached for the categorization of issue records as non-bugs. As a result of this initial work, it appears that a system that automatically alerts users or software developers whether an issue record is caused by a software bug, is possible.

Keywords: Bug Reports, Information Retrieval, Data Analysis, Data Mining.

1 Giriş

Yazılım sistemlerinin kalitesini artırmak için, kullanıcıların ve testçilerin karşılaştıkları yazılım hatalarını raporlayabilmeleri için çeşitli yazılım hata izleme sistemleri kullanılmaktadır. Ancak büyük yazılım sistemlerinde her gün yüzlerce kayıt açılabilmektedir. Anvik ve arkadaşları Mozilla uygulamasında her gün 300 kadar yeni hata kaydı açıldığını, bu kadar çok kaydın yazılımcıların çözebilecekleri sayıların çok üstünde olduğunu raporlamışlardır [1].

Yazılımcılara yardımcı olması amacıyla hata kayıtlarının otomatik olarak sınıflandırılması konusunda çeşitli deneysel çalışmalar yapılmıştır. Hata kayıtlarının kim tarafından çözülmesi gerektiği [2], ne kadar sürede çözülebileceği [3], geçerli olup olmadıkları [4], şiddeti [5], mükerrer olup olmadıkları [6] gibi konularda otomatik olarak tahminleme yapılması denenmiş ve elde edilen sonuçlar raporlanmıştır. Bu çalışmalarda geleneksel yaklaşım, hata kayıtlarında yer alan özet ve açıklama alanlarında geçen metin verisi kullanılarak, ilgili veri içinde geçen terimlerin tf-idf endeksi ile gösterimi ve makine öğrenmesi teknikleri ile otomatik olarak sınıflandırılmasıdır.

Softtech 1000 kişinin üzerinde nitelikli iş gücü ile, 2 Teknokent ofisi ve 2 Ar-Ge merkezi olmak üzere toplam 7 farklı yerleşkede, yenilikçi projeler üreten bir Ar-Ge şirketidir. Ana faaliyet alanı finans ve bankacılık olmakla birlikte, çeşitli yelpazede ürünler sunmaktadır. Softtech de üretim ortamında oluşan hata kayıtlarını yönetmek için hata kaydı izleme sistemlerini kullanmakta ve bu süreçte yazılımcıların hayatlarını kolaylaştıracak çeşitli yaklaşımlara ihtiyaç duymaktadır.

Bu çalışmada Softtech'in üretim ortamında çözümlediği olay kaydı veri seti kullanılmıştır. Üretim ortamında her gün 400 civarı kayıt açılmakta ancak son zamanlarda bu kayıt sayılarında artış eğilimi olduğu gözlenmektedir. Çalışmadaki amacımız bu artış eğilimini analiz etmek ve aslında yazılım hatası olmayan olay kayıtları otomatik olarak tahminleyebilen veya öneride bulunabilecek bir sistemin mümkün olup olmadığını ortaya koymaktır. Dolayısıyla çalışma ile aşağıdaki araştırma sonuçlarına yanıt aranmıştır:

- Endüstriyel vakaya ait olay kayıtlarının içinde yazılım hatası olmadığı yönünde değerlendirilebilecek kayıtların oranı nedir?
- İlgili kayıtları otomatik olarak başarılı bir şekilde sınıflandırabilir miyiz?

Bildirinin geri kalan kısmı şu şekilde organize edilmiştir: Bölüm 2'de ilgili çalışmalar aktarılmaktadır, Bölüm 3'te analiz ve sınıflandırma konusunda uyguladığımız yöntem sunulmaktadır, Bölüm 4'te yapılan deneyler ve elde edilen sonuçlar tartışılmaktadır, Bölüm 5'te yapılan deneysel çalışmanın geçerliliğine dair tehditler irdelenmektedir ve Bölüm 6'da sonuçların özeti sunulmakta ve gelecek olası çalışmalar hakkında bilgi verilmektedir.

2 İlgili Çalışmalar

Antoniol ve arkadaşlarının çalışmaları [4], bir olay kaydının yazılım hatası kaynaklı olup olmadığının sınıflandırılması yönünde yapılan ilk çalışmalardandır. Bu çalışmada Mozilla, Eclipse ve JBoss projelerinden elde ettikleri 1800 adet olay kaydını el ile yazılım hatası kaynaklı olan ve olmayan olarak sınıflandırmışlardır. Karar Ağacı, Naive Bayes ve Lojistik Regresyon algoritmalarını kullanarak %82-'ye varan bir doğrulukla otomatik olarak sınıflandırma yapabilmişlerdir. Bu çalışmada aynı zamanda, hata kayıtlarındaki çeşitli özniteliklerin sınıflandırma performansına etkisini araştırmışlardır. Zhou ve arkadaşları [10] aynı verisetini

kullanarak metin madenciliği ve veri madenciliği tekniklerini birleştirerek daha iyi sonuçlar elde edebildiklerini ortaya koymuşlardır.

Herzig ve arkadaşları ise olay kaydı takip sistemlerinde tutulan kayıtların hatalı sınıflandırılmasından kaynaklanan maliyeti detaylı bir şekilde çalışmışlardır [11]. Beş farklı açık kaynak olay kaydı takip sisteminden elde ettikleri 7401 kapalı kaydın gözden geçirilmesinin 90 gün sürdüğünü ve bu kayıtların üçte birinin aslında yazılım kaynaklı olmadığını, dolayısıyla hatalı sınıflandırıldığını raporlamışlardır.

Pandey ve arkadaşları da [12] üç farklı açık kaynak olay kaydı takip sisteminden elde ettikleri kayıtlarla benzer bir çalışma yapmış ve kayıtları yazılım kaynaklı olan ve olmayan şeklinde sınıflandırmışlardır. RF (Random Forest) ve DVM (Destek Vektör Makinesi) gibi algoritmalarla projeye bağlı olarak %75-83 doğruluk oranlarına ulaşabildiklerini görmüşlerdir. Çalışmada kayıtları ön işlemden geçirirken ilgili sınıflarda en sık geçen terimleri seçmişlerdir.

3 Yöntem

Önerilen yöntemin amacı; vakaya ait kayıtların nedenlerini ortaya koyarak yazılım hatası kaynaklı olup olmadıklarını tespit etmek, sonrasında ise bu kayıtları başarılı bir şekilde sınıflandırmanın mümkün olup olmadığını ortaya koymaktır. Yöntem bölümünde öncelikle kayıtların yazılım hatasından kaynaklandığını ne şekilde ortaya koyduğumuz, sonrasında ise doğal dilde yazılmış olay kayıtlarının sözcük çantası modeli ile temsili ve sınıflandırılması anlatılmıştır.

3.1 Veri Seti

Çalışmada 2019 yılının ilk beş ayındaki "Tamamlandı" ve "İade Edildi" statüsündeki olay kayıtları kullanılmıştır. Bu kayıtların üzerinde olay kaydı olup olmadıklarına dair bir alan bulunmamaktadır. Ancak yazılımcılar kayıtları "Tamamlandı" veya "İade Edildi" seçeneği ile kapatırken, sebep kategorisi alanını, kendilerine sunulan seçeneklerden birisini seçerek doldurmaktadır. Bu alandaki bilgi kullanılarak kayıtlar, yazılım kaynaklı olan veya olmayan olarak sınıflandırılmıştır. Öğrenme aşamasında Ocak-Nisan ayları arasında açılan kayıtlar, test aşamasında ise Mayıs ayında açılan kayıtlar kullanılmıştır.

3.2 Yazılım Hatasından Kaynaklanmayan Olay Kayıtlarının Tespiti

Olay kayıtları yazılımcılar tarafından kapatılırken sebep kategorilerinden birisi seçilmektedir. Kayıt işlem yapılarak kapatılıyorsa, "Tamamlandı" statüsüne getirilmekte, işlem yapılmadan iade ediliyorsa "İade Edildi" statüsüne getirilmektedir. Her iki durumda da olası sebep kategorilerinden birisinin seçilmesi gerekmektedir.

Öncelikle, olay kaydı adetleri, ilgili sebep kategorileri ve oluşturuldukları ay bilgisine göre çıkarılmış ve hangilerinde aylık bazda artış eğilimi olduğu ortaya konmuştur. Sonrasında, kapalı olan kayıtlardan sebep kategorisi "Kullanıcı bilgi

eksikliği", "Olay olarak nitelendirilmemelidir", "Operasyonel talep", "Geliştirme talebi" vb. olanlar (1) "Yazılım Hatası Kaynaklı Olmayan", kalan diğer kayıtlar da (2) "Yazılım Hatası Kaynaklı" olarak gruplandırılmıştır. Sınıflandırma çalışmasında olay kaydının bu iki sınıftan hangisine ait olduğu tahminlenmiştir.

3.3 Olay Kayıtlarının Sınıflandırması

Yazılım hata raporlarının sınıflandırılması için hatanın anlatıldığı özet ve açıklama alanlarındaki metin verisi girdi olarak kullanılır. Metin verisinde istatistiksel olarak anlam ifade eden terimlerin tespiti için öncelikle metinler kelimelere bölünür (tokenization) ve durak kelimeler (stop words) ayıklanır. Daha önceki çalışmalarda yazılım hata sınıflandırmasında kelime köklerine inmenin etkisinin düşük olduğu raporlandığından [7] bu çalışmada da kök bulma işlemi yapılmamıştır.

Elde edilen kelimeler kullanılarak n-gramlar hesaplanır. Bu çalışmada 2-gram kullanılmıştır, 2-gram art arda gelen 1'li ve 2'li kelime dizinleridir. Sonuç olarak elde edilen her bir kelime veya kelime dizini *terim* olarak adlandırılmaktadır. İlgili terimlerin her bir yazılım hata kaydı için ne kadar anlam ifade ettiğinin anlaşılması için *tf-idf* endeksi kullanılır; *tf* terim frekansını, *idf* ise ters doküman frekansını ifade eder [8]. Bir hata kaydında geçen her bir terim için *tf-idf* değeri aşağıdaki şekilde hesaplanır:

$$tf(t,o) = \frac{t \text{ teriminin } o \text{ olay kaydında görülme sayısı}}{o \text{ olay kaydındaki toplam terim sayısı}} \tag{1}$$

$$idf(t,O) = log(\frac{O \text{ havuzundaki toplam olay kaydı sayısı}}{t \text{ terimini içeren olay kaydı sayısı}}) \tag{2}$$

$$tf - idf(t,o,O) = tf(t,o) \times idf(t,O) \tag{3}$$

Destek Vektör Makineleri (DVM) metin madenciliğinde sıklıkla kullanılan ve başarılı sonuçlar veren bir algoritmadır. DVM algoritması ile, farklı iki sınıf arasında maksimum marj bırakan bir düzlem oluşturularak sınıflandırma hataları en aza indirgenmeye çalışılır. Marj, iki sınıfın düzleme en yakın noktalarından düzleme olan uzaklıklarının toplamıdır [9]. Çalışmada amacımız farklı algoritmaları kıyaslamak olmayıp bilinen bir algoritma ile başarılı sonuçlar elde edip edemediğimizi ortaya koymaktır. Bu nedenle Pandey ve arkadaşlarının yakın zamanda yaptıkları çalışma sonucunda başarılı sonuçlar elde ettiklerini raporladıkları DVM algoritması seçilmiştir [12]. Çalışmamızda öğrenme aşamasında 10-kat çapraz doğrulama yöntemi kullanılarak model oluşturulmuştur.

Sonuçları değerlendirmek için kesinlik, duyarlılık ve f-skor metrikleri kullanılmıştır. Kesinlik (precision), bir sınıf için doğru tahmin edilmiş kayıt sayısı, o sınıf için yapılan toplam tahmin sayısına bölünerek bulunmaktadır. Duyarlılık (recall) ise bir sınıf için doğru tahmin edilmiş kayıt sayısı, o sınıfa ait gerçek kayıt sayısına bölünerek bulunmaktadır. F-skor, kesinlik ve duyarlılığın harmonik ortalamasıdır. Metriklerin hesaplanma yöntemleri aşağıda verilmiştir [8].

$$kesinlik(s) = \frac{s \text{ sınıfı için doğru yapılan tahmin adedi}}{s \text{ sınıfı için yapılan tahmin adedi}} \tag{4}$$

$$duyarlılık(s) = \frac{s\ sınıfı\ için\ doğru\ yapılan\ tahmin\ adedi}{veri\ setinde\ s\ sınıfına\ ait\ kayıt\ adedi} \tag{5}$$

$$f - skor(s) = \frac{2 \times kesinlik(s) \times duyarlılık(s)}{kesinlik(s) + duyarlılık(s)} \tag{6}$$

4 Vaka Çalışması Sonuçları

Bu bölümde öncelikle sebep kategorilerine göre kayıt adetleri aylık bazda tespit edilerek, toplam adetlerin artışında etkili olan konular ortaya konulmuştur. Sonrasında ise ilgili kayıtlar ve bu kayıtların içerdiği metin bilgisi kullanılarak, yazılım hatası kaynaklı olmayan kayıtlar için tahminleme çalışması yapılmış ve sonuçlar aktarılmıştır.

4.1 Sebep Kategorilerine göre Kayıt Adetleri

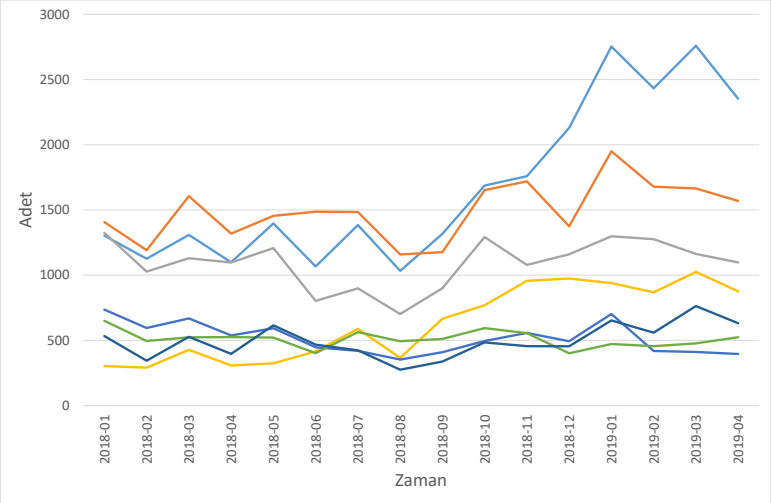
Şekil 1'de "Tamamlandı" statüsündeki kayıtların sebeplerine göre ("Kullanıcı bilgi eksikliği", "Operasyonel Talep" vb.) aylık bazda dağılımları, Şekil 2'de ise "İade Edildi" statüsündeki kayıtların sebeplerine göre ("Kullanıcı bilgi eksikliği", "Operasyonel Talep" vb.) dağılımları yine aylık bazda verilmiştir. (Şekillerde ilgili çizgilerin hangi sebep kategorisine ait oldukları firmaya ait özel bilgi olduğu için belirtilmemiştir.)

"Tamamlandı" statüsündeki kayıtlar için 2018 Kasım ayından itibaren artış eğiliminde olan ve en fazla adede sahip kayıtlar, "Kullanıcı bilgi eksikliği/talebi" sebep kategorisi seçeneği ile kapatılmış kayıtlardır. "İade Edildi" statüsündeki kayıtlar içinse diğer tüm kayıt tiplerinden çok daha fazla adede sahip kayıtlar, "Olay kaydı olarak nitelendirilmemelidir" seçeneği ile kapatılan kayıtlar olup, son dönemde bu tipteki kayıt adetlerinde de artış gözlenmektedir.

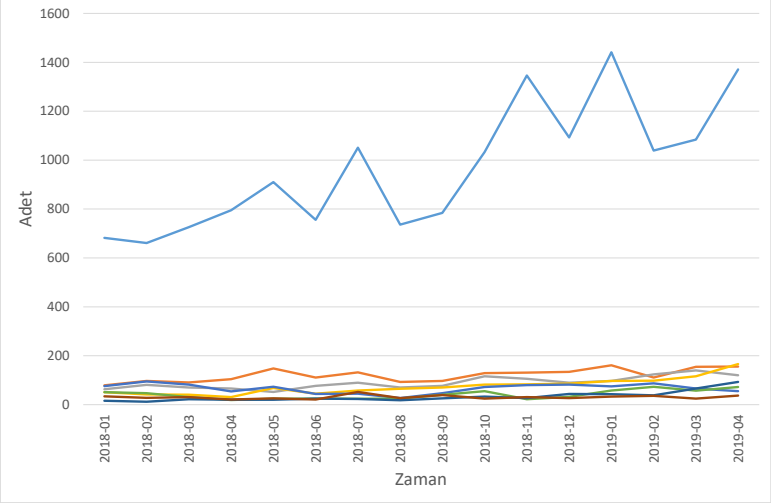
Her iki sebep kategorisi de aslında yazılım kaynaklı olmayan kayıtlarda son dönemde artış olduğunu işaret etmektedir. Bu durum, olay kayıtlarının kaynak nedeninin otomatik olarak tespit edilebilmesinin önemini ortaya koymaktadır. Yazılımsal kaynaklı olmayan kayıtların otomatik olarak tespit edilebilmesi ve kullanıcılara bilgi amaçlı olarak gösterilmesinin, yazılımcıların verimliliğinin artırılmasında ve genel olarak olay kaydı yönetim sürecinin iyileştirilmesinde olumlu katkısı olacağı öngörülmüştür.

4.2 Yazılım Hatası Kaynaklı Olmayan Kayıtların Tahminlenmesi

Çalışma için 2019 yılında açılan kayıtlardan statüsü "Tamamlandı" veya "İade Edildi" olanlar kullanılmıştır. Bölüm 2.1'de anlatıldığı şekilde, sebep kategorisi alanına bakılarak, kayıtlar "Yazılım Hatası Kaynaklı" ve "Yazılım Hatası Kaynaklı Değil" olarak iki sınıfa ayrılacak şekilde işaretlenmiştir. Veriye ait özet bilgi Tablo 1'de verilmiştir. Yazılım hatasından kaynaklanmayan kayıtların adedinin toplam kayıt adedine oranının oldukça yüksek olduğu görülmektedir (% 44).



Şekil 1. "Tamamlandı" statüsündeki kayıtların sebep kategorilerine göre aylık adetleri.



Şekil 2. "İade Edildi" statüsündeki kayıtların sebep kategorilerine göre aylık adetleri.

Tablo 1. Elde edilen veriye ait özet bilgi.

Zaman aralığı	Toplam Adet	Yazılım hatasından kaynaklanmayan kayıt adedi
Ocak 2019 - Mayıs 2019	41.930	18.385

DVM algoritması ile yapılan tahminleme çalışması sonucunda Tablo 2'de verilen kesinlik, duyarlılık ve f-skor değerlerine ulaşılmıştır. Sonuçlar, bu endüstriyel vakada, geleneksel bir yaklaşım kullanılarak dahi, yüksek başarı ile yazılım hatasından kaynaklanmayan kayıtların tahminlenebildiğini göstermiştir.

Tablo 2. Tahminleme çalışması sonuçları.

Sınıf	Kesinlik	Duyarlılık	F-Skor
Yazılım hatasından kaynaklanan	0,88	0,78	0,83
Yazılım hatasından kaynaklanmayan	0,72	0,84	0,77

5 Geçerliliğe Tehditler

Çalışma yürütülürken yazılımcıların sebep kategorisi alanını doldurdukları bilginin geçerli olduğu, hatalı bir seçim yapmadıkları varsayılmıştır. Olay kaydının yazılım hatasından kaynaklanıp kaynaklanmadığı bilgisine, yazılımcıların yaptıkları bu seçimden yola çıkılarak ulaşılmıştır.

Çalışma endüstriyel vakaya ait tek bir veri seti ile yürütülmüştür. Başka vaka çalışmalarında farklı sonuçların elde edilmesi mümkündür. Çalışmada amacımız olay kaydı sistemlerinde aslında yazılım hatası kaynaklı olmayan kayıtların çokluğunu ve bilinen bir teknik ile başarılı tahminleme sonuçları elde edilebildiğini ortaya koymaktır, bu nedenle metin madenciliği konusunda iyi sonuçlar verdiği bilinen DVM algoritması seçilmiştir. Çalışma yapılırken algoritmanın parametreleri ayarlanmamış, varsayılan parametre değerleri ile sonuçlar elde edilmiştir.

6 Sonuç ve Gelecek Çalışmalar

Bu çalışma ile öncelikle olay kaydı adetlerinde artışa neden olan kayıtların kaynak nedenleri tespit edilmiştir. Bu artışta yazılımcılar tarafından, kullanıcı bilgi eksikliği veya olay olarak nitelendirilmemesi gerektiği düşünülen kayıtların etkili olduğu görülmüştür. Yazılımcılar tarafından seçilen kayıt nedenleri gruplanarak incelendiğinde, yazılım hatasından kaynaklanmayan kayıtların oldukça yüksek adetlere ulaştığı görülmüştür. Yapılan tahminleme çalışmasında yazılım hatası kaynaklı olmayan kayıtların başarılı bir şekilde tahminlenebildiği ortaya konmuştur. Gelecek çalışmalarda bu tip kayıtların önüne geçebilmek için alınabilecek aksiyonların ve sistemde yapılabilecek otomasyonların değerlendirilmesi düşünülmektedir.

Kaynakça

1. Anvik, J., Hiew, L., Murphy, G. C. (2005). Coping with an open bug repository. In Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange (pp. 35-39). ACM.
2. Anvik, J., Hiew, L., Murphy, G. C. (2006). Who should fix this bug?. In Proceedings of the 28th international conference on Software engineering (pp. 361-370). ACM.
3. Weiss, C., Premraj, R., Zimmermann, T., Zeller, A. (2007). How long will it take to fix this bug?. In Fourth International Workshop on Mining Software Repositories (MSR'07: ICSE Workshops 2007) (pp. 1-1). IEEE.
4. Antoniol, G., Ayari, K., Di Penta, M., Khomh, F., Guéhéneuc, Y. G. (2008). Is it a bug or an enhancement?: a text-based approach to classify change requests. In CASCON (Vol. 8, pp. 304-318).
5. Menzies, T., Marcus, A. (2008). Automated severity assessment of software defect reports. In 2008 IEEE International Conference on Software Maintenance (pp. 346-355). IEEE.
6. Bettenburg, N., Premraj, R., Zimmermann, T., Kim, S. (2008). Duplicate bug reports considered harmful... really?. In 2008 IEEE International Conference on Software Maintenance (pp. 337-345). IEEE.
7. Murphy, G., Cubranic, D. (2004). Automatic bug triage using text categorization. In Proceedings of the Sixteenth International Conference on Software Engineering and Knowledge Engineering.
8. Manning, C., Raghavan, P., Schütze, H. (2010). Introduction to information retrieval. Natural Language Engineering, 16(1), 100-103.
9. Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In European conference on machine learning (pp. 137-142). Springer, Berlin, Heidelberg.
10. Zhou, Y., Tong, Y., Gu, R., Gall, H. (2016). Combining text mining and data mining for bug report classification. Journal of Software: Evolution and Process, 28(3), 150-176.
11. Herzig, K., Just, S., Zeller, A. (2013). It's not a bug, it's a feature: how misclassification impacts bug prediction. In Proceedings of the 2013 international conference on software engineering (pp. 392-401). IEEE Press.
12. Pandey, N., Sanyal, D. K., Hudait, A., Sen, A. (2017). Automated classification of software issue reports using machine learning techniques: an empirical study. Innovations in Systems and Software Engineering, 13(4), 279-297.

İki Güncel Dinamik Topluluk Takibi Yönteminin Karşılaştırmalı Bir Çalışması

Arzum Karataş[1] , Serap Şahin[1]

[1] Izmir Institute of Technology, Izmir, Turkey

arzumkaratas@iyte.edu.tr, serapsahin@iyte.edu.tr

Özet. Gerçek dünya ağları doğaları gereği dinamiktirler ve sanal ortamlarda çoğunlukla dinamik çizgelerle temsil edilirler. Bu dinamik ağlardaki verilerin analizi kriminoloji, politika, sağlık, reklamcılık ve sosyal ağlar vb. gibi pek çok alanda karar destek sistemleri için değerli bilgilerin elde edilmesine katkıda bulunur. Toplulukların takibi, ağda bulunan toplulukların dinamizmini ve eğilimlerini analiz etmek, anlamak ve bu toplulukların yakın geleceklerinin kestirimi için çok önemlidir. Bu verilerin başarılı bir şekilde analiz edilmesi halinde yazılım mühendisliği araçları ve karar destek sistemleri son kullanıcılar için daha sağlıklı sonuçlar üretir. Bu çalışmada, seçtiğimiz iki önemli ve güncel yöntemi doğruluğu, algoritmik karmaşası ve genel özellikleri bakımından karşılaştırmalı olarak inceledik. Biz bu çalışmada topluluk takibi üzerine geliştirilmiş ve her basamakta tespit edilen topluluk bilgisini içeren sentetik veri kümeleri kullandık.

Anahtar Kelimeler: Dinamik ağlar, Toplulukların tespiti, Topluluk gelişimlerinin takibi, Topluluk takibi.

A Comparative Analysis of Two Most Recent Dynamic Community Tracking Methods

Arzum Karataş[1] , Serap Şahin[1]

[1] Izmir Institute of Technology, Izmir, Turkey

arzumkaratas@iyte.edu.tr, serapsahin@iyte.edu.tr

Abstract. Real world networks are intrinsically dynamic, and they are mostly represented by dynamic graphs in virtual world. Analysis of these dynamic network data can give valuable information for decision support systems in many

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

domains in criminology, politics, health, advertising and social networks etc. Community tracking is important to analyze and understand the dynamics of the group structures and predict the near futures of communities. With a successful analysis of these data, software engineering tools and decision support systems can produce more successful results for end users. In this study, we present a comparative study of two important and recent community tracking methods in terms of accuracy, algorithmic complexity and their characteristics. We use a benchmark dataset which have ground truth community information detected each time step as a test bed.

Keywords: Dynamic networks, Community detection, Community tracking.

1 Introduction

Real world networks such as social, biological, telecommunication etc. are dynamic and they generally evolve gradually due to the interaction of their members. They are mostly represented by dynamic networks constructed from a series of static networks ordered in time.

Community tracking is one of the important problems in the domain of Network Science because it helps analyzing information diffusion [1] and observing change of group dynamics etc. Researchers are interested in examining evolution of communities to study the evolutionary trend and predict the future structure of the network for support decision systems in many areas such as public health (to discover dynamics of certain groups susceptible to a disease) [2], criminology (to identify criminal groups that spread or support criminal ideas and activities like terrorism) [3] and politics (to observe influences of political ideologies on some social group over time). Similarly, community tracking provides valuable information for support decision systems for targeted marketing [4] and recommendation systems [5].

A comparative study of six selected methods that follow independent community detection and matching approach for tracking communities in evolving (or dynamic) social networks is done by He et al. [6] in 2017. However, He et al. [6] only test four of the methods such as Greene et al.'s [7], Takaffoli et al.'s [8], Brodka et al.'s [9] and Tajeuna et al.'s [10] methods between 2010 and 2015. In this study, they compare these methods for overlapping and disjoint communities with two specific measures (e.g. Average Pearson Correlation [10] and Proportion of Nodes Persisting [10]) to evaluate the quality of the tracked communities. As a result of their study, they state that all the approaches compared track communities very well. However, they neither compare accuracy nor do complexity analysis for these methods which are important for usage of community tracking software tools by end users. As an additional note, all studies under reference [6] work on a pool of static data, which is divided into many time slices. The selected methods aim to detect event relations among communities of different time steps. We also trace the existence of new methods in literature after the work of He et al. [6] till June 2019. Finally, we select Greene et al.'s method [7] and Tajeuna et al.'s [10] method for our study and we explain the reasons why we select them in Section 2.2.

The main contributions of this paper are to (1) detailed examination of important two community tracking methods (e.g., Greene et al.'s [7] from 2010 and Tajeuna et al.'s [10] from 2015) including their methodological steps, space and running time complexity analysis, (2) comparison of these methods in terms of their characteristics and accuracy rates and (3) at last evaluate their pros and cons.

The rest of the paper is organized as follows. In Section 2, we give some preliminary information about community tracking, and detailed information about Greene et al.'s [7] and Tajeuna et al.'s [10] method. In Section 3, we present our experimental works. In Section 4, we evaluate the experimental results and we close our paper by giving our final thoughts.

2 Selected Methods for Tracking Community Evolutions on Dynamic Networks

2.1 Concept Definitions

We use a graph $G_t=(V_t, E_t)$ for representing a static network (e.g., actual snapshot of the dynamic network) where V_t stands for the set of members (e.g., vertices) and E_t the set of connections (e.g., edges). We represent a dynamic network as an ordered sequence of static networks like $G=\{G_1, G_2, \ldots, G_s\}$where s is the number of static networks, which built the dynamic network.

Real world networks inherently contain a community structure inside where a community is a subset of members of each time graph G_t is densely connected inside rather than the rest of the network. There can be number of *k* communities belong to same G_t. Community detection reveals underlying group structure of the current time graph G_t like $C_t = \{C_t^1, C_t^2, \ldots, C_t^k\}$ where each community is $C_t^i \in C_t$ and $C_t^i=(V_t, E_t)$ is a subset of C_t.

Given two communities C_1 and C_2 are regarded as similar when they share members more than a given similarity threshold in our work. The most popular similarity is Jaccard similarity (e.g., intersection over union) for community matching.

An evolving community is denoted by ordered sequence of tracked communities. Each community residing on this tracking chain indicates the status of evolving community at a specific time step. For example, $S_{C^1} = \{C_{t1}^1, C_{t2}^1, \ldots, C_{t5}^1\}$ represents the evolution of community C^1 from time step t_1 to t_5. Therefore, community tracking problem produces evolution chains for the communities on dynamic networks.

As time goes on, an evolving community structure may change because of arrival or departure of members and connections. Therefore, an evolving community may be *growing* with arrival of new members while it may be *shrinking* with the departure of existing members or continues with nearly same members. In the same way, a community may *split* into different communities or several communities may *merge* and form a new community over time. We also observe that a new community is formed (e.g., *birth)*, or a community *dissolves* by losing its members. Also, some communities can evolve non-consecutively, which means that a community observed at time t is

observed at time t+2 or later time steps instead of time t+1.

2.2 Community Tracking Methods

We select two recent community tracking methods that follow independent community detection and matching approach to compare. This approach detects communities on each time step and then match them among different time steps. We select Greene et al.'s method [7] because it is one of the outstanding works in the area and its source code is open to public. Thus, it is readable and modifiable. We select Tajeuna et al.'s method [10], because it has most advanced capabilities (i.e., it is the only one able to detect all community evolution events among existing community tracking methods and the other capabilities are seen in Table 6). Therefore, we present them hereunder as selected related works. Note that, also both methods need pre-calculated communities of dynamic network at each time step and we use the main variables below when we do complexity analysis of the methods: *number of time steps (t)* , *number of nodes (n)*, *number of communities (c), average size of communities per time step(avg_com_size)* and *number of dynamic communities (d)*.

Greene et al.'s Method. A pseudocode for Greene et al.'s method [7] is provided in Fig. 1. The authors regard each dynamic community as a timeline(D_i). They initialize a set of timelines($\mathbb{D}$) with initial communities ($\mathbb{C}_i$) on first time step (Step 1).The latest communities on these timelines are called as fronts (F_i) of respective timeline. They read each time step communities (C_t^j) and try to match the current time step communities with fronts (Step 2). To do matching between each community of this time step and fronts, they build a map for tracking the place of each node in the fronts and hold an array to compute and count intersections among them. They compute Jaccard similarity between fronts and current time step communities in constant time by using this array (Step 2.2.1) and if similarity is higher than similarity threshold λ, they add the compared step community to the matched community's timeline (Step 2.2.2). If there is no match, they append the set of timelines with a new dynamic community containing currently matched community (Step 2.2.3). Then, they update the fronts (Step 2.3).

As it seen from Table 1, the most time-consuming task in this method is building and filling a map for tracking the place of each node in the fronts in Step 2.2.1 for preparation of matching process. Note that traversing nodes for each time step takes at most $O(t \times n)$ computation time and adding them into a map requires at most $O(log\ n)$ computation time. Therefore, time complexity of the method is $O(t \times n \times log\ n)$; in overall. The most space-consuming steps are Step 2.1. and Step 2.2.1. Therefore, the method needs memory space as $O(n + (d \times avg\_com\_size) \times long)$ type for storing nodes at each time step and nodes at front communities.

1. Initialize $\mathbb{D}$ by creating a new D_i for initial communities $C_1^x \in \mathbb{C}_1$ and initialize each F_i with C_1^x
2. For each time step t>1
    2.1 Read $\mathbb{C}_t$ from community files
    2.2 For each $C_t^j \in \mathbb{C}_t$
        2.2.1 Match C_t^j with F_i where sim $(C_t^j, F_i) > \lambda$
        2.2.2 If $match(C_t^j, F_i) = true$, then append D_i with (C_t^j)
        2.2.3 Else append $\mathbb{D}$ with a new D containing (C_t^j)
        2.2.4 End for
    2.3. Update fronts
3. End for

Fig. 1. A pseudocode for Greene et al.'s method

Table 1. Complexity values of each step at pseudocode in Fig. 1.

Step	Time Complexity	Space Complexity
1	$O(n)$; traverse all nodes of communities at t=1	$O(n \times long\ int)$; store all nodes of communities at t=1
2.1	$O(t \times n)$; read all nodes in the network	$O(n \times long\ int)$; store all nodes of communities at t
2.2.1	$O(t \times n \times log\ n)$; read all nodes of communities at *t* and put them into a map	$O(d \times avg\_com\_size \times long\ int)$; store all nodes in the front + $O(d \times int)$; store intersection of a community in the array with fronts where array indices refer to front indices
2.2.2	$O(t \times \frac{1}{\lambda})$; process every matched community (e.g., their count equals at most $\frac{1}{\lambda}$) for each t	$O(\frac{1}{\lambda} \times int)$; holds the indices of matched communities
2.2.3	$O(c \times t)$; append each unmatched community (e.g., their count equals c at worst case) at each *t* to $\mathbb{D}$	$O(c \times int)$; store indices of communities to be appended.
2.3	$O(d)$; traverse all dynamic communities at worst case	$O(d \times int)$; store the front indices

Tajeuna et al.'s Method. A pseudocode for the method is shown in Fig. 2. The authors first build a binary matrix(A) between nodes and communities to represent memberships of nodes at Step 1. They create a Burt matrix(B) where $B = A^T \times A$ to define the number of common nodes between communities in Step 2; keeping mind we use Eigen library residing http://eigen.tuxfamily.org website for fast matrix multiplication. They transform B into a transition matrix to calculate the transition probabilities between communities by normalizing entries of B in Step 3. They track communities in Step 4. For this reason, they hold a set S and initialize S with currently tracked community (C_i) to hold evolution chain of at Step 4.1. They check mutual transition (e.g. sim()) between each community with most recent time (t_x) in S($C_{t_x}^k$)and each community at time step $t_j(C_{t_j}^m)$ and check Jaccard similarity(JS) between $C_{t_j}^m$ and C_i. If both mutual transition and JS is greater than the similarity threshold λ, $C_{t_j}^m$ is added to evolution chain, S at Step 4.2.1.1. Mutual transition between two communities is sum of harmonic means of each entries of correspondent transition matrix rows. Jaccard similarity is directly computed correspondent entries of normalized Burt matrix.

1. Build A[][] to represent community memberships of nodes
2. Build B[][] where $B = A^T \times A$
3. Normalize B[][] in place
4. For each community C_i to be tracked
   - 4.1 Initialize S with C_i, t_i= time step of C_i , $t_j = t_i + 1$
   - 4.2 While $t_j \leq t$
     - 4.2.1 For each community $C_{t_x}^k$ with t_x last time in S
       - 4.2.1.1 *If* $sim(C_{t_x}^k, C_{t_j}^m) > \lambda$ *and* $JaccardSim(C_i, C_{t_j}^m) > \lambda$ *then* append S with $C_{t_j}^m$
     - 4.2.2 End for
     - 4.2.3 $t_j = t_j + 1$
   - 4.3 End while
5. End for

Fig. 2. A pseudocode of Tajeuna et al.'s method

In Table 2, complexity values of each step in Fig. 2 are seen. Specifically, the most time-consuming task is similarity calculation between tracked communities and prospective next time step communities at Step 4.2.1.1. This calculation is done for all communities at next time step ($O(c)$) for each tracked community ($O(c \times t)$) and the similarity calculation cost between two community is $O(c \times t)$. Therefore, the running time complexity of the method is $O((c \times t)^2 \times c)$. The space complexity of the work is $O((c \times t)^2 \times (double))$ because of the storage of Burt matrix.

Table 2. Complexity values of each step at pseudocode in Fig. 2.

Step #	Time complexity	Space Complexity
1	$O(t \times n)$; traverse the nodes of all communities	$O(c \times t \times (int))$; store time information of communities $+ O(n \times t \times (double))$; Eigen stores non-zero coefficients of A
2	$O(t \times c \times avg\_com\_size)$; Eigen multiplies only non-zero coefficients of A^T and A	$O((c \times t)^2 \times (double))$; store correlations between all communities (matrix B)
3	$O((c \times t)^2)$; traverse each cell of matrix B	No need extra space because normalization of B is done in place
4	$O((c \times t)^2 \times c)$; traverse all communities to be tracked ($O(c \times t)$) and for each of them traverse each next time step community $C_{t_j}^m$ ($O(c)$) and traverse each cell of normalized Burt matrix for similarity calculation ($O(c \times t)$).	$O\big(c \times (string)\big)$; to store names of $C_{t_j}^m$ communities $O((c \times t) \times (string))$; store community names to be tracked and list of tracked communities

3 Experiments & Results

3.1 Data Sets, Assumptions and Environment for Experiments

In this section, we test Greene et al.'s method and Tajeuna et al.'s method on a dynamic benchmark dataset. This dataset is generated by Greene et al. The dataset is constructed from four different synthetic graphs where each of them contains five static networks (e.g., five time steps) with 15000 nodes to simulate nonconsecutively evolving communities and includes six different community evolution events such as birth, death, growing, shrinking, merge and split. The dataset can be found online at http://mlg.ucd.ie/dynamic.

- The '*Intermittent'* dataset contains the nonconsecutively evolving communities.
- The '*BirthDeath'* dataset is created to simulate form and dissolution of communities. Greene et al. create birth of 40 additional communities by transferring nodes of other existing communities and randomly remove 40 existing communities along the network.

- The '*ExpandContraction'* dataset is created to simulate growing and shrinking of communities. Greene et al. make growing and shrinking for randomly chosen 40 communities by 25% of their previous sizes.

- The '*MergeSplit'* dataset is created to simulate merge of two existing communities and splitting of an existing community into two new communities. Greene et al. introduce 40 split and 40 merge events.

We design the following experiment to compare their accuracies and running speeds under the same conditions. For the equality of similar steps of both algorithms (e.g. detection of existing communities at each step); we use Louvain algorithm [11] and we remove the communities that have less than 3 nodes for both approaches.

We download executable and source code of Greene et al.'s method [7] and use for experimental results. But, since neither source code nor executable of Tajeuna et al.'s method is open to public, we implement their method with same programming language (C++) with Greene et al.'s. We use a laptop with Core i7 2.30 GHz CPU and 8GB memory with our testbeds. Note that similarity threshold is taken as 0.1 for both methods because Greene et al. use this value in their experiment.

3.2 Results of Experiments

It is possible to evaluate the quality of detected communities by using Normalized Mutual Information (NMI) [12], if the ground-truth community information is already known. NMI measures accuracy of detected community structure by comparing them with the ground-truth structure and it produces a real value in [0,1]. The higher NMI values show that community detection done is better. The measurements are seen in Table 3 and Table 4 for each benchmark dataset.

The NMI values recorded in Table 3 are values with 92% accuracy rate. For the work of Tajeuna et al.[10], the NMI values reside in Table 4 with 98% accuracy rate. We calculate them via an NMI calculation software [13], by feeding of ground truth and detected community information for each step.

Table 3. NMI values for Greene et al.'s method[7]

Time-steps	BirthDeath NMI	ExpandContraction NMI	Intermittent NMI	MergeSplit NMI
t=1	0,88	0,88	0,88	0,88
t=2	0,93	0,94	0,91	0,94
t=3	0,92	0,93	0,94	0,92
t=4	0,92	0,96	0,93	0,91
t=5	0,94	0,96	0,92	0,88

In the light of accuracy values that we obtain, we can say that Tajeuna et al.'s method performs better than Greene et al.'s method in terms of accuracy.

Table 4. NMI values for Tajeuna et al.'s method[10]

Timesteps	BirthDeath NMI	ExpandContraction NMI	Intermittent NMI	MergeSplit NMI
t=1	0,996729	0,998628	0,99493	0,995583
t=2	0,995638	0,99618	0,987166	0,974798
t=3	0,995151	0,981107	0,98827	0,944636
t=4	0,991624	0,977629	0,992919	0,934176
t=5	0,994112	0,974361	0,989547	0,905201

4 Discussion of the Methods and Conclusion

Table 5. Overview of the methods

Capabilities		**Greene et al.'s method (2010)**	**Tajeuna et al.'s method (2015)**
1	Threshold setting	Manually set	Automatically set
2	Tracking	Consecutive & partially nonconsecutive	Consecutive & nonconsecutive
3	Missing Event type	Continue event type is not defined.	All event types are defined.
4	Merge events	Identification is limited to two communities.	There is no limitation. n communities can merge.
5	Split events	A community can only be split into two communities.	There is no limitation. A community can be split into m communities
6	Space Complexity	$O(n + (d \times avg\_com\_size) \times (long))$	$O((c \times t)^2 \times (double))$
7	Time Complexity	$O(t \times n \times log\ n)$	$O((c \times t)^2 \times c))$
8	Accuracy on the benchmark dataset	92 %	98%

In this study; we have presented a comparative study of two recent community tracking methods. Greene et al.'s method [7] is published in 2010 and Tajeuna et al.'s [10] method is published in 2015. Both methods focus on dynamically evolving data networks, to analyze their community relations on static and time sliced data pools. We test them on ground truth benchmark datasets in terms of accuracy and do their

complexity analysis. Hereunder, Table 5 summarizes a comparative analysis for selected criteria between both methods.

Criteria 1 & 2 & 3 & 4 & 5 are satisfied by Tajeuna et. al.'s method with higher accuracy. Threshold setting is automatically tuned according to the average number of nodes shared among all detected communities. Which is requirement overall high accuracy on continuing analyses among communities for long time series. Tracking of each community with all possible event types on non-consecutive time steps is important.

Criteria 6 & 7: Running times obtained from our experimental study are matched with time complexity analysis. Greene et. al.'s method should be preferred in very fast computation with predetermined similarity threshold requirement, but results are less accurate. On the other hand, Tajeuna et al.'s method should be preferred in the need of high accuracy and automatic similarity threshold requirements, but it needs high computation time with sufficient memory space.

In future, we will focus on tracking communities on real world networks and predicting community evolution events.

References

1. S. Lin, Q. Hu, G. Wang, and S. Y. Philip: Understanding community effects on information diffusion. pp. 82-95. In Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining(PAKDD), pp. 82-95, Springer, Ho Chi Minh City, Vietnam (2015)

2. D. A. Luke, and J. K. Harris: Network analysis in public health: history, methods, and applications. Annu. Rev. Public Health 28, 69-93(2007)

3. A. Calvó-Armengol, and Y. Zenou: Social networks and crime decisions: The role of social structure in facilitating delinquent behavior. International Economic Review 45(3), 939-958(2004)

4. D. Kempe, J. Kleinberg, and É. Tardos: Maximizing the spread of influence through a social network. pp. 137-146. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 137-146, ACM, Washington DC, USA(2003)

5. M. Zanin, P. Cano, J. M. Buldú, and O. Celma: Complex networks in recommendation systems. pp 120-124. In Proceedings of second International Conference on Computer Engineering and Applications, World Scientific Advanced Series In Electrical And Computer Engineering(WSEAS), pp 120-124, Acapulco, Mexico(2008)

6. Z. He, E. G. Tajeuna, S. Wang, and M. Bouguessa: A Comparative Study of Different Approaches for Tracking Communities in Evolving Social Networks. pp. 89-98. In Proceedings of 2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp 89-98, Tokyo, Japan(2017)

7. D. Greene, D. Doyle and P. Cunningham: Tracking the evolution of communities in dynamic social networks. pp. 176-183. In Proceedings of International conference on Advances in social networks analysis and mining (ASONAM), pp. 176-183,IEEE,Odense, Denmark(2010)

8. M. Takaffoli, J. Fagnan, F. Sangi, and O. R. Za¨ıane: Tracking changes in dynamic information networks. Pp 94-101. In Proceedings of International Conference on Computational Aspects of Social Networks (CASoN), pp. 94–101, Salamanca, Spain(2011)

9. P. Brodka, S. Saganowski, and P. Kazienko: Ged: the method for group evolution discovery in social networks. Social Network Analysis and Mining 3(1), 1–14(2013)

10. E. G. Tajeuna, M. Bouguessa, and S. Wang: Tracking the evolution of community structures in time-evolving social networks. pp 1-10. In Proceedings of International Conference on Data Science and Advanced Analytics (DSAA), pp. 1–10, Paris, France(2015)

11. V. D. Blondel, J.-L. Guillaume, R. Lambiotte and E. Lefebvre E.:Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment 2008(10), 10008(2008)

12. L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas: Comparing community structure identification. Journal of Statistical Mechanics: Theory and Experiment 2005(09), P09008(2005)

13. A. Lancichinetti, S. Fortunato and J. Kertesz:Detecting the overlapping and hierarchical community structure in complex networks. New Journal of Physics, 11(3), 033015(2009)

Kullanıcı Merkezli Tasarımın Televizyon Etkileşimine Uygulanması

Evrim Akgündüz[1,2], Yasemin Topaloğlu[1]

[1] Ege Üniversitesi, İzmir, Türkiye
[2] Vestel, Manisa, Türkiye
evrimakgunduz@gmail.com, yasemin.topaloglu@ege.edu.tr

Özet. Hızla gelişen teknoloji ile birlikte geleneksel televizyon sistemleri, yerini akıllı televizyonlara bırakmıştır. Akıllı televizyonlarda kullanıcılar yalnızca yayın akışını izlemek yerine; yayın akışını durdurabilir, kaydedebilir, televizyondan oyunlar oynayabilir ve hatta internet üzerinden e-posta gönderme, alışveriş yapma gibi daha pek çok kişisel aktiviteyi gerçekleştirebilmektedir. Ortaya çıkan "akıllı" özelliklerle birlikte geleneksel uzaktan kumanda ile TV etkileşimi yetersiz kalmıştır. Akıllı TV'lerde geleneksel uzaktan kumandaya alternatif olarak jest (harekete duyarlı) ile kontrol, ses ile kontrol ya da mobil uzaktan kumanda uygulaması gibi TV kontrol yöntemleri geliştirilmiştir. Bu son etkileşim türleri, TV üreticileri tarafından araştırılıp geliştirilmekte ve kullanıcıların geleneksel kumandadan farklı olarak televizyonu kontrol etmelerine olanak tanımaktadır. Bu çalışmada televizyon etkileşimi için bir mobil uzaktan kumanda uygulaması prototipinin kullanıcı merkezli tasarım yaklaşımı temel alınarak geliştirilmesi adımları sunulmuş olup kullanıcı arayüzü kalitesini arttırmak amacıyla kullanılabilirlik değerlendirmesi gerçekleştirilmiştir. Ortaya çıkan tasarım sorunlarına çözüm önerileri sunulmuştur.

Anahtar Kelimeler: Kullanılabilirlik, TV, Uzaktan Kumanda, Kullanıcı Merkezli Tasarım, Mobil Uygulama

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Applying User-Centered Design to Television Interaction

Evrim Akgündüz[1,2], Yasemin Topaloğlu[1]

[1] Ege University, Izmir, Turkey
[2] Vestel, Manisa, Turkey
evrimakgunduz@gmail.com, yasemin.topaloglu@ege.edu.tr

Abstract. In this study, how to interact in today's television systems, is discussed with examples from the literature in terms of usability, and an approach to implement user-centered design principles for interaction in television systems is proposed. With the rapidly developing technology, traditional television systems have been replaced by modern television systems. In modern television systems, users do not only watch the broadcast stream; they can stop broadcasting, record, play games on television, and even perform other personal activities such as sending mail, shopping on the internet. Together with the smart features, the TV interaction with the traditional remote control became inadequate. Modern TVs replace the traditional remote control; alternative TV control methods such as control with gesture, control with voice or mobile remote control application were developed. These novel interaction types are researched and developed by TV manufacturers and allow users to control the television as opposed to the conventional controller. In this study, the steps of developing a mobile remote control prototype for television interaction based on user - centered design approach are presented and usability evaluation is performed to improve user interface quality. Suggestions for solutions to emerging design problems are presented.

Keywords: Usability, TV, Remote Controller, User-Centered Design, Mobile Application

1 Giriş

Hızla gelişen teknoloji ile birlikte geleneksel televizyon sistemleri yerini akıllı televizyonlara bırakmıştır. Akıllı TV'ler, ev eğlencesi için giderek daha önemli bir çoklu ortam cihazı haline gelerek TV programları başta olmak üzere birçok medya türüne ve aynı zamanda oyunlara, internete, sosyal ağ siteleri gibi servislere erişim sağlayan bir platformdur [1].

Akıllı televizyonlarda kullanıcılar yalnızca yayın akışını izlemek yerine; yayın akışını durdurabilir, kaydedebilir, televizyondan oyunlar oynayabilir ve hatta internet üzerinden e-posta gönderme, alışveriş yapma gibi daha pek çok kişisel işlerini gerçekleştirebilir. Televizyonların geleneksel işlevlerinin dışındaki işlevselliğinin artmasına bağlı olarak televizyonlarda etkileşimin önemi artmıştır. Televizyonlarla ilgili literatürdeki çalışmalar incelendiğinde; çoğunlukla arayüz tasarımının nasıl

olması gerektiğine odaklanılmıştır. Ancak son zamanlarda yalnızca televizyon arayüzü veya tasarımının nasıl olması gerektiği gibi konular yerine; özellikle televizyon ile etkileşim türlerinin de incelenmesi önemli bir konu haline gelmiştir [2]. Son yıllarda geleneksel uzaktan kumandaya alternatif olarak jest etkileşimleri, mobil uzaktan kumanda uygulama etkileşimleri ve ses etkileşimleri gibi yeni alternatifler sunulmuştur. Bu etkileşim türleri, TV üreticileri tarafından araştırılıp geliştirilmekte ve kullanıcıların geleneksel kumandadan farklı olarak televizyonu kontrol etmelerine olanak tanımaktadır [3]. Yeni gelişen televizyon etkileşim türlerinde de kullanılabilirlik önemli bir bileşendir [4].

Bu çalışmanın amacı, mobil uzaktan kumanda kullanılabilirliğini geliştirmek için kullanıcı merkezli tasarım yaklaşımı temel alınarak televizyon etkileşimi için mobil uygulama prototipi geliştirilmesi adımları sunulmuştur. Mobil uzaktan kumanda kullanıcısının performansını, tercihini değerlendiren belirli görevler ve senaryolar oluşturulup nesnel ve öznel verilerin ölçümleri yapılmıştır. Bu ölçümlerden elde edilen bulgulara göre tasarıma dair öneriler sunulmuştur.

Bildirinin ikinci bölümünde etkileşim tasarımı ve kullanılabilirliğin ilişkisi tartışılmıştır. Üçüncü bölümde literatürdeki televizyon sistemlerinin etkileşim yöntemleri hakkında kısaca bilgi verilerek, çalışmada seçilen etkileşim yöntemi açıklanmıştır. Dördüncü bölümde kullanıcı merkezli tasarım ile geliştirme süreci adımları tanıtılmıştır. Beşinci bölümde geliştirilen prototip uygulama üzerinde yapılan değerlendirme ile ilgili bilgi verilerek altıncı bölümde çalışmanın sonuçları tartışılmıştır.

2 Etkileşim Tasarımı ve Kullanılabilirlik

İnsan-Bilgisayar Etkileşimi (İBE), bilgisayar teknolojisinin insan işlerini ve faaliyetlerini nasıl etkilediğinin incelenmesidir. Bazen "etkileşim tasarımı" veya "kullanıcı merkezli tasarım" olarak adlandırılır [16].

Kullanılabilirlik, en yalın tanımıyla bir yazılımın ya da insan etkileşimi gerektiren her türlü ürünün kolay kullanılabilmesi ve kolay öğrenilebilmesidir [7].

Kullanılabilirlik, ISO'ya (Uluslararası Standartlar Enstitüsü) göre bir uygulamada belirlenen işlerin, hedeflenen kullanıcı kitlesi tarafından etkili ve verimli bir şekilde ve sonuç olarak memnun kalarak gerçekleştirilmesi olarak tanımlanmaktadır [5].

Nielsen [6] ise kullanılabilirliği kullanıcıların bir sistemi kabulünü temel alarak tanımlamış ve kullanılabilirliğin çok fazla bileşeni olduğunu ancak kolay öğrenebilirlik, etkin kullanım, hatırlanabilirlik, hatalar ve kullanıcı memnuniyeti bileşenleri ile ilişkili olduğunu belirtmiştir.

Kullanılabilirliğin amacı; kullanım kalitesini sağlamaktır.Kullanılabilirlik değerlendirmesi, etkileşimli bir sistemin kullanılabilirliğini, işlevselliğini ve kabul edilebilirliğini test eder. Değerlendirme teknikleri uzman tabanlı yaklaşım ve kullanıcı tabanlı yaklaşım olarak iki grupta toplanmıştır. Uzman tabanlı yaklaşım; analitik yöntemler, inceleme yöntemleri, model tabanlı yöntemleri kapsarken kullanıcı tabanlı yaklaşım ise; deneysel yöntemler, gözlem yöntemleri, sorgu yöntemlerini kapsamaktadır [20].

Kullanıcı tabanlı yaklaşıma göre gözlem yönteminde; değerlendirici sesli düşünme, protokol izleme gibi tekniklerle, kullanıcıların eylemlerini izler ve kaydeder. Böylece sistemin karar süreçleri veya tutumları hakkında fikir edinilmiş ve kullanıcıların gereksinimlerini ne kadar iyi karşılandığı belirlenmiş olunur [20].

Kullanıcı tabanlı değerlendirmede kullanılan sorgu yöntemleri ise; röportaj ya da anket teknikleri ile kullanıcıya doğrudan arayüz hakkında soru sormaya dayanır. Kullanıcının bir sisteme bakışının ayrıntılarını ortaya çıkarmak için yararlı olabilir. Bir sistemin kullanıcı gereksinimlerini nasıl karşıladığını bulmanın en iyi yolunun 'kullanıcıya sormak' olduğunu belirten felsefeyi somutlaştırır. Kullanıcı gereksinimleri ve görevleri hakkında bilgi toplamak için değerlendirmede ve daha yaygın olarak kullanılabilirler.

Çalışmada kullanıcı tabanlı değerlendirme yaklaşımı uygulanmış olup, katılımcılardan verilen çeşitli görevleri yerine getirmeleri istenmiştir. Sesli düşünme ve sorgulama tekniği kullanılarak mobil uzaktan kumanda uygulaması hakkında kullanıcı yorumları alınmıştır [20].

3 Televizyon Sistemlerinde Etkileşim ve Kullanılabilirlik

Gelişen teknoloji ile birlikte geleneksel TV'ler yerini, kullanıcıya yapacağı işlemlerde özgürlük tanıyan, sadece yayın akışını izlemek yerine oyun oynama veya e-ticaret sitesi üzerinden işlemler yapmaya olanak tanıyan akıllı TV'lere bırakmıştır. Akıllı TV'lerle ilgili kullanılabilirlik çalışması bulunmaktadır. Bu çalışmalar çoğunlukla kullanılabilirlik değerlendirmesi için kurallar tanımlama ve arayüz tasarımı hakkında yol gösterici olma özelliği taşımaktadır. Bununla birlikte TV etkileşim yöntemlerine ait kullanılabilirlik çalışmaları da yapılmıştır [2][3][13].

Andrés Solano ve ark. [8] çalışmalarında, kullanılabilirlik değerlendirmelerine yardımcı olmak amacıyla Nielsen'ın sezgiselinе dayanan sezgisel bir yöntem önerilmiştir. Önerilen sezgisel değerlendirme kuralı; tasarım-estetik, esneklik-navigasyon, hatalar-yardım olmak üç ana çerçeveyi kapsamakta olup TV kullanılabilirlik kontrol listesi sunmaktadır.

Lyn Pemberton ve Richard N. Griffiths [9] ise televizyonlarda kabul edilebilirliği ölçmek için kullanılabilirlik laboratuvarının kurulmasının gerekli olduğunu, ancak tek başına kullanılabilirlik laboratuvar sınamalarının yeterli olmadığını belirterek anket gibi sorgulama yöntemiyle daha sağlıklı değerlendirme yapılacağına varmışlardır.

Marianna Obrist ve ark. [10] yaptıkları çalışmada ise kullanılabilirlik kavramının, evrensel kullanılabilirlik açısından değerlendirilmesi gerekliliğini düşünürek; "yaşlı insanlar navigasyon odaklı bir TV'yi nasıl algılar ve yorumlar?" ana odağı üzerinde durmuşlardır. Sonuç olarak TV uygulamalarının tasarımında, birçok yaşlı insanın maruz kaldığı bilişsel, duyusal ve fiziksel kısıtlamaların (örneğin görme bozukluğu) dikkat edilmesi gereken noktalar olduğu konusunda önemli çıktılar elde edilmiştir.

Andrea Ingrosso ve ark. [2], uzaktan kumanda ile akıllı TV arasındaki etkileşime odaklanmıştır. Akıllı TV'de e-ticaret uygulaması ayrıntılı olarak değerlendirilmiş ve uzaktan kumandanın yalnızca ikonları seçmesi açısından değil, aynı zamanda ekrandaki komutlarla tutarsızlık riski taşıması yönüyle de ergonomik olmadığı tespit edilmiştir. Çalışmada bu unsurların akıllı TV ile etkili ve verimli bir şekilde etkileşim

kurulmasına engel olduğu ve jest / ses tanıma sistemlerinin kullanılmasının e-ticaret gibi uygulamalarda etkileşimi iyileştirebileceği belirtilmiştir.

Jan Bobeth ve ark. [13], özellikle yaşlı kullanıcılara odaklanarak serbest el hareketlerinin performansını ve kabulünü araştırmıştır. Çalışmada, TV'nin metin girilen bir ekranda imleci kontrol etmek için doğrudan izlenen el hareketlerinin aktarılmasının uzaktan kumandaya göre en iyi performansı sağladığını ve kullanıcılar tarafından tercih edildiği ortaya çıkmıştır.

Jorge Abreu ve ark. [3], sesli komut ile TV etkileşimi mobil bir uygulama, uzak alan mikrofonu ve mikrofonlu uzaktan kumanda aracılığıyla olmak üzere kullanılabilirlik açısından değerlendirilmiştir. Çalışmaya göre, sesli komut ile TV kontrolü düşük dijital okuryazarlığı olan kullanıcılar ve özellikle fiziksel sınırlamaları olan kullanıcılar için kullanım kolaylığı sunmuştur. Yapılan detaylı literatür taraması sonuçlarına göre mobil uzaktan kumanda uygulamasının KMT prensibi ile gerçekleştirimine ve mobil uzaktan kumanda için kullanılabilirlik değerlendirilmesine rastlanılmamıştır.

4 Kullanıcı Merkezli Tasarım ile Televizyon Etkileşimi

Kullanıcı merkezli tasarım (KMT-User Centered Design-UCD), kullanıcıları sistemlerin tasarlanma aşamasına dahil etmeye odaklanan bir felsefe ve yöntem için genel bir terimdir [14]. KMT, geliştirme sürecinde kullanıcıları değişime zorlamak yerine, kullanıcıların dünyasına uyan bir yazılım üretmek amacıyla kullanıcıların ihtiyaçlarını, isteklerini ve kısıtlamalarını dikkate alan yinelemeli bir yöntemdir. [15].

KMT yaklaşımını tercih etmemizdeki önemli nedenlerden birisi TV etkileşim aracı geliştirirken "kullanıcı" temelli bir tasarım yaklaşımıyla kullanıcının ihtiyaçlarıyla örtüşen aynı zamanda da yazılım kalitesinden ödün vermeyen bir etkileşim aracı oluşturulması amaçlanmıştır. Şekil 1.'de gösterildiği gibi KMT genel süreci, küçük yinelemeli adımlarla uygulamanın döngüsel değerlendirilmesine dayanmaktadır.



Şekil 1. Kullanıcı Merkezli Tasarım

Mobil etkileşim uygulamasına "Neden ihtiyacımız var?" sorusuna kullanıcılardan aşağıdaki yanıtlar verilmiştir.

1. Evde bulunan birden çok elektronik eşyaya ait uzaktan kumandaların zaman zaman birbirleriyle karıştırılabiliyor olması.
2. Uzaktan kumandaların beklenmedik zamanlarda pillerinin bitebiliyor olması.
3. Cep telefonunun artık hayatımızın bir parçası haline gelip her yerde yanımızda taşıyor olmamız.

4.1 Kullanım İçeriğinin Belirlenmesi

Kullanıcı merkezli tasarım için öncelikle kullanım içeriğinin belirlenmesi gerekmektedir. Bu aşamada; kullanıcıların kim olduğu, ürünü neden ve hangi ortamda kullanacakları belirlenmiştir.

Çalışmamızda, 20-35 yaş aralığında 3 erkek ve 4 kadın, 35-50 yaş aralığında 3 erkek ve 5 kadın, 50+ yaş aralığında 3 erkek ve 2 kadın olmak üzere farklı yaş gruplarından 20 katılımcı ile gerçekleştirilmiştir. Katılımcıların tümü akıllı TV, akıllı cep telefonu ve bilgisayar kullanıcısı olmalarına rağmen 20-35 yaş grubu katılımcıların tamamı, 35-50 yaş grubundan 3 katılımcı, 50+ yaş grubunun hiçbirisi olmak üzere toplam katılımcıların %50'si daha önce mobil uzaktan kumanda uygulaması kullanıcısı bireylerdir.

Mobil uzaktan kumanda prototipi Android işletim sistemi için geliştirilmiştir.

4.2 Gereksinimlerin Belirlenmesi

İçerik belirlendikten sonra, gereksinimleri tanımlamak gerekmektedir. Bu aşamada tasarımcıların işini kolaylaştırmak ve ürünü başarılı kılmak için "*persona*" kullanılmıştır. Böylece kullanıcıların sistemi nasıl kullanacakları ve ne bekleyebilecekleri konusunda net bir fikir elde edilip rahatlıkla gereksinimler toplanılmış ve tasarım etkinlikleri yürütülmüştür.

Her bir persona belirli bir kullanıcı grubunu temsil eder ve oluşturulan her kullanıcı grubunun hedefleri, davranışları, motivasyonları hakkında bilgiler sunmaktadır. Çalışmada 20-35 yaş, 35-50 yaş ve 50+ yaş grubu özelliklerini yansıtan persona oluşturulmuştur.

- **Hedef_1:** Kullanıcılar cep telefonu ile akıllı TV etkileşimini başarı bir şekilde gerçekleştirmek istiyor.
- **Davranış:** Uygulama üzerinden akıllı TV'ye komut gönderip kanal değişikliği, ses değişikliği, TV açma-kapama, menü işlevlerini gerçekleştirebilme.
- **Motivasyon:** Kullanıcılar tek bir cihaz olarak cep telefonu ile TV'yle etkileşim hedeflerine ulaşmak istiyorlar.

- **Hedef_2:** Kullanıcılar birden çok elektronik eşyaya ait uzaktan kumandaların birbirleriyle karıştırılabilmesinin önlemek amacıyla tek bir cihaz kullanmak istiyor.
- **Davranış:** Uygulama üzerinden akıllı TV'ye komut gönderip kanal değişikliği, ses değişikliği, açma kapama, menü işlevlerini gerçekleştirebilme.
- **Motivasyon:** Diğer **e**lektronik aletlerin kumandalarının TV kumandası ile karıştırılması gibi durumlara maruz kalmak istemiyorlar.
- **Hedef_3:** Kullanıcılar uzaktan kumandaların beklenmedik zamanlarda pillerinin bitmesi durumda TV ile etkileşimin hala kolaylıkla gerçekleştirebilmek istiyor.
- **Davranış:** Uygulama üzerinden akıllı TV'ye komut gönderip kanal değişikliği, ses değişikliği, açma kapama, menü işlevlerini gerçekleştirebilme.
- **Motivasyon:** Uzaktan kumandanın pili bitmese bile TV'yi hala rahatlıkla kumanda edebilmek istiyorlar.

4.3 Tasarım ve Gerçekleştirimi

Tasarım sürecinin başlarında doğru yönde olup olmadığımızı doğrulamak amacıyla ve geri bildirim almak için katılımcılar ile Şekil 2.'te gösterilen kâğıt modeller üzerinde çalışma yapılmıştır. Böylece tasarımın uygunluğunu, çok fazla çalışma yapmadan belirlememize yardımcı olmuştur. Ayrıca etkileşim tasarımı için kâğıt modeller kullanmak, tasarımın hızlı bir şekilde kullanıcı isteklerine göre değiştirilebilmesine olanak vermiştir.

Kâğıt modellerden alınan geri bildirimler ışığında prototipin arayüzü, Şekil 3.'te gösterildiği gibi geliştirilmiştir. Prototipin ilk aşamasında geleneksel kumandada bulunan; TV ses arttırıp-azaltma, TV kanal değiştirme, TV'yi sessize alma, TV açma-kapama, TV menü tuşları eklenmiştir.



Şekil 2. Kâğıt Modeller



Şekil 3. Prototipe ait arayüz görselleri

4.4 Kullanılabilirlik Değerlendirmesi

Değerlendirme, ürün hakkında geri bildirim almak için çok önemli bir adımdır. Alınan geri bildirimlere göre uygulamanın kullanıcı ihtiyacını karşılayıp karşılamadığı belirlenir ve ihtiyaca yönelik eksikler varsa tasarım hedefleri ve kullanıcı ihtiyaçları karşılanana kadar yinelemeli olarak devam eder.

Çalışmada kullanıcı ihtiyaçlarının karşılanıp karşılanmadığını belirlemek için sesli düşünme ve sorgulama tekniği olmak üzere kullanıcı tabanlı değerlendirme gerçekleştirilmiştir. Kullanılabilirlik değerlendirmesi sürecinde katılımcılardan çeşitli görevleri yerine getirmeleri istenmiştir. Bu görevler, kullanıcıların geleneksel kumanda üzerinde sıklıkla gerçekleştirdikleri aktiviteler dikkate alınarak belirlenmiştir. Tablo 1.'de katılımcılarının mobil uzaktan kumanda uygulamasında kullanılabilirlik değerlendirmesi sırasında yerine getirdiği görevleri göstermektedir.

Mobil uzaktan kumanda uygulaması ile akıllı TV'de görevleri tamamlamak için 20 katılımcıyla kullanılabilirlik testi yapılmıştır. Her test oturumu sırasında, kullanıcılar görevleri gerçekleştirirken ne yaptığını, neden yaptığını ve nasıl hissettiğini açıklamaları istenmiştir. Böylece yüksek sesle düşünme tekniği uygulanmıştır. Bir test sorumlusu ve gözlemci, kullanılabilirlik testi oturumu sürecinde kullanıcılar görevleri yerine getirirken meydana gelen olayları, etkinlikleri ve yüksek sesle düşünme hakkında notlar almışlardır.

Görevlerin tamamlanmasının ardından katılımcılardan uygulama hakkında kişisel fikirlerini almak için tek tek görüşleri toplanmıştır.

Tablo 1. Katılımcılar için görev tablosu

GÖREV	TANIMLAMA
1	TV'ye başarı ile bağlanma
2	TV kapama
3	TV açma
4	Kanal değiştirme
5	Ses arttırma
6	Ses azaltma
7	TV'yi sessize alma
8	Menü tuşu ile menü ekranının açılması ve bu ekrandan çıkış yapılması
9	Menü tuşu ile menü ekranının açılması ve bu ekrandayken geri tuşuna basılması

5 Bulgular

Kullanılabilirlik değerlendirmesine göre katılımcıların verilen görevleri yerine getirme süreleri Tablo 2. , Tablo 3. ve Tablo 4.'te görülmektedir.

20-35 yaş aralığı katılımcı değerlendirme ve görüşme bulguları;

- **TV'ye bağlanma** işleminde zorluk yaşamadıklarını belirtmişlerdir.
- **TV kapama ve açma görevi** için açma kapama butonun her sayfada bulunmasının erişilebilirlik açısından daha iyi olacağı belirtilmiştir.

- **Kanal değiştirme görevi** esnasında zorlukla karşılaşılmadığı ancak uygulamada açılan ilk sayfa olarak gelmesinin daha kullanışlı olacağı belirtilmiştir.
- **Ses arttırma ve azaltma görevi** esnasında bir zorlukla karşılaşılmadığı belirtilmiştir.
- **TV sessize al görevi** sırasında tuşun yine açma kapama tuşu gibi her sayfada olmasının daha kolay bulunacağı belirtilmiştir.
- **Menü tuşuna tıklayıp geri ve çıkış görevi** için bir zorlukla karşılaşılmadığı belirtilmiştir.

Tablo 2. 20-35 yaş görev tamamlama süresi

20-35 yaş aralığında katılımcı	Görev	Ortalama Görev Tamamlama Süresi(sn.)	Toplam Süre(sn.)
	Görev_1	6.6	
	Görev_2	4.1	
	Görev_3	2.3	
	Görev_4	7.8	
	Görev_5	3.2	46.6
	Görev_6	2.9	
	Görev_7	10.3	
	Görev_8	4.3	
	Görev_9	4	

35-50 yaş aralığı katılımcı değerlendirme ve görüşme bulguları;

- **TV'ye bağlanma** işleminde zorluk yaşamadıklarını belirtmişlerdir. Ancak bağlanma süreci hakkında bilgilendirmenin olmasının daha iyi olacağı belirtilmiştir.
- **TV kapama ve açma görevi** için açma kapama butonun her sayfada bulunmasının erişilebilirlik açısından daha iyi olacağı belirtilmiştir.
- **Kanal değiştirme görevi** esnasında zorlukla karşılaşılmadığı ancak uygulamada açılan ilk sayfa olarak gelmesinin daha kullanışlı olacağı belirtilmiştir.
- **Ses arttırma ve azaltma görevi** esnasında bir zorlukla karşılaşılmadığı belirtilmiştir.
- **TV sessize al görevi** sırasında tuşun yine açma kapama tuşu gibi her sayfada olmasının daha kolay bulunacağı belirtilmiştir.
- **Menü tuşuna tıklayıp geri ve çıkış görevi** için bir zorlukla karşılaşılmadığı belirtilmiştir.

Tablo 3. 35-50 yaş görev tamamlama süresi

35-50 yaş aralığında katılımcı	Görev	Ortalama Görev Tamamlama Süresi(sn.)	Toplam Süre(sn.)
	Görev_1	8	
	Görev_2	5.2	

	Görev_3	3.1	
	Görev_4	9.6	
	Görev_5	4.3	54.2
	Görev_6	3.2	
	Görev_7	12.5	
	Görev_8	4.3	
	Görev_9	4	

50+ yaş aralığı katılımcı değerlendirme ve görüşme bulguları;

- **TV'ye bağlanma** işleminde zorluk yaşamadıklarını belirtmişlerdir.
- **TV kapama ve açma görevi** esnasında katılımcılar görme bozuklarının olmasına bağlı olarak açma kapama butonun daha büyük olabileceğini belirtmişlerdir. Ayrıca kullanım esnasında açma kapama butonun her sayfada bulunmasının kolay kullanım sağlayacağı belirtilmiştir.
- **Kanal değiştirme görevi** esnasında kanal arttırma ve azaltma butonunu bulmakta zorlanmışlardır. Ayrıca yine butonların daha büyük olmasının daha iyi olacağı belirtilmiştir.
- **Ses arttırma ve azaltma görevi** esnasında bir zorlukla karşılaşılmadığı ancak rahat tıklama için büyük bir şekilde konumlandırılmasının daha iyi olacağı belirtilmiştir.
- **TV sessize al görevi** sırasında tuşun yine açma kapama tuşu gibi her sayfada olmasının daha kolay bulunacağı belirtilmiştir.
- **Menü tuşuna tıklayıp geri ve çıkış görevi** yerine getirmede bir zorlukla karşılaşılmadığı belirtilmiştir.

Tablo 4. 50+ yaş görev tamamlama süresi

50+ yaş aralığında katılımcı	Görev	Ortalama Görev Tamamlama Süresi(sn.)	Toplam Süre(sn.)
	Görev_1	10.1	
	Görev_2	6.4	
	Görev_3	4.3	
	Görev_4	11.5	
	Görev_5	5.1	64.3
	Görev_6	4.7	
	Görev_7	13.6	
	Görev_8	4.5	
	Görev_9	4.1	

Her bir görev yerine getirilmeden önce başlangıç noktası olarak uygulamanın açıldığı ilk ekran temel alınarak süreler hesaplanmıştır. Görevleri yerine getirme toplam süreleri 20-35 yaş aralığında 46.6 sn, 35-50 yaş aralığında 54.2 sn ve 50+ yaş katılımcı grubunda 64.3 sn sürmüştür.

Elde edilen bulgulara göre TV'ye ait özelliklerden sessize alma butonu ve kanal değiştirme butonlarının konumlandırıldığı sayfa görev tamamlama sürelerinde etkili olmuştur. 20-35 yaş aralığı katılımcıların %100'ünün mobil uzaktan kumanda kullanım deneyiminin olması, 35-50 yaş aralığı katılımcıların %37.5'sinin mobil

uzaktan kumanda kullanmış olması, 50+ yaş aralığı katılımcıların hiçbirinin daha önce mobil uzaktan kumanda kullanmamış olması ortalama görev tamamlama süresini etkilemiştir.

20-35 yaş aralığında katılımcıların %28.5'i Android cep telefonu kullanmıyor olup ortalama görev süresini etkilemiştir. 35-50 yaş ve 50+ yaş aralığı katılımcıların tamamı Android cep telefonu kullanmakta olup arayüze aşinadırlar.

6 Sonuç

Akıllı TV'ler, ev eğlencesi için giderek daha önemli bir çoklu ortam cihazı haline gelerek TV programları başta olmak üzere birçok medya türüne ve aynı zamanda oyunlara, internete, sosyal ağ siteleri gibi servislere erişim sağlamaktadır. Akıllı TV'lerin geleneksel işlevlerinin dışındaki işlevselliğinin artmasına bağlı olarak televizyonlarda etkileşimin önemi artmıştır. Ancak etkileşim yöntemlerinin gelişen teknoloji ile kullanıcılar tarafından tercih edilmesini etkenlerinden birisi kullanılabilirliktir.

Bu çalışmada, mobil uzaktan kumanda uygulaması prototipi KMT prensipleri uygulanarak geliştirilmiş ve katılımcılarla kullanılabilirlik değerlendirmesi gerçekleştirilmiştir. Kullanılabilirlik değerlendirme sürecinden sonra, görev tamamlanma süresi ölçümlerine dayanarak toplanan verileri ve katılımcılardan alınan öznel yorumları analiz ettik. Sonuçlar, KMT prensibi uygulanarak değerlendirilen mobil uzaktan kumanda uygulamasının birkaç kullanılabilirlik sorununa sahip olduğunu ve bunların kullanıcıların çeşitli süreçlerdeki performansını etkilediğini göstermektedir.

Bu çalışmanın sonuçları, mobil uzaktan kumanda uygulamasını kullanırken kullanıcıların odak noktalarını göstermektedir. Ayrıca, mobil uzaktan kumanda uygulamasının kötü tasarım özellikleri de ortaya çıkarılmıştır. Çalışmada elde edilen sonuçlara göre, mobil uzaktan kumandasının arayüzünü tasarlarken aşağıdaki ilkelere önem verilmelidir:

1. Butonlar kolay tıklanabilecek büyüklükte tasarlanmalıdır.
2. Uygulama üzerinde yapılan işlemlerin durumunu göstermek için bilgilendirici geri bildirim verilmelidir.
3. TV işlevlerine ait kanal değiştirme, ses artırma-azaltma gibi birincil işlevlerin butonları kolayca bulunabilecekleri şekilde konumlandırılmalıdır.
4. TV açma-kapama ve sessize alma gibi işlemlere ait butonlar kumanda uygulamasına ait tüm sayfalarda bulunmalıdır.

Mobil uzaktan kumanda uygulamasının kullanılabilirliğini geliştirmek için bu birkaç tasarım ilkesini öneriyoruz. Bu çalışmadan elde edilen sonuçların daha fazla kullanıcı merkezli mobil uzaktan kumanda tasarımında faydalı olacağını umuyoruz.

Kaynakça

1. Choi, Y. M., & Li, J. (2016). Usability Evaluation of a New Text Input Method for Smart TVs. *Journal of Usability Studies*, (s. 110-123).
2. Ingrosso, A., Volpi, V., Opromolla, A., Sciarretta, E., & Medaglia, C. M. (2015). UX and Usability on Smart TV: A Case Study on a T-commerce Application. *International Conference on HCI in Business*, (s. 312-323).
3. Abreu, J., Beça, P., Santos, R., Cardoso, B., Fernandes, S., & Rodrigues, A. (2018). Voice Interaction on TV: Analysis of natural language interaction models. *Proceedings of the XIX International Conference on Human Computer Interaction*, (s. Article No. 8).
4. Bevan, N. (1995). Usability is Quality of Use. *Proceedings of the 6th International Conference on Human Computer Interaction.*
5. ISO 9241-11:2018 . (2018). Ergonomics of human-system interaction -- Part 11: Usability: Definitions and concepts.
6. Nielsen, J. (1993). Usability Engineering. Academic Press, Boston, USA.
7. Bevan, N. (1995). Measuring usability as quality of use. *Software Quality Journal*, (s. 115-130).
8. Solano, A., Collazos, C. A., Rusu, C., & Roncagliolo, S. (2011). Usability Heuristics for Interactive Digital Television. *The Third International Conference on Advances in Future Internet (AFIN 2011).*
9. Pemberton, L., & Griffiths, R. N. (2003). Usability Evaluation Techniques for Interactive Television.
10. Obrist, M., Beck, E., Bernhaupt, R., & Tscheligi, M. (2007). Focusing on Elderly: An iTV Usability Evaluation Study with Eye-Tracking. *Interactive TV: a Shared Experience, 5th European Conference, EuroITV 2007*, (s. 66-75). Amsterdam.
11. Chorianopoulos, K., & Spinellis, D. (2006). User interface evaluation of interactive TV: a media studies perspective. *Universal Access in the Information Society*, (s. 209-218).
12. Solano, A., Collazos, C. A., Rusu, C., & Merchán, L. (2013). Evaluating the Usability of Interactive Digital Television Applications. *10th International Conference on Information Technology: New Generations (ITNG 2013).*
13. Bobeth, J., Schmehl, S., Kruijff, E., Deutsch, S., & Tscheligi, M. (2012). Evaluating performance and acceptance of older adults using freehand gestures for TV menu control. *EuroITV '12 Proceedings of the 10th European Conference on Interactive TV and Video*, (s. 35-44).
14. Abras, C., Maloney-Krichmar, D., & Preece, J. (2004). User-centered design. *W. Encyclopedia of Human-Computer Interaction*, (s. 445-456). Bainbridge.
15. ISO 9241-210. (2010). *Ergonomics of human-system interaction -- Part 210: Human-centred design for interactive systems.*
16. Dix, A. Human-Computer Interaction., (s. 1327-1331).
17. *Wholesale – Buy China Wholesale Products on DHgate.com*. (tarih yok). https://www.dhgate.com/product/wholesale-remote-control-original-for-grundig/400876333.html adresinden alındı
18. *Pinterest*. (tarih yok). https://tr.pinterest.com/pin/347692033701480143/ adresinden alındı
19. *Amazon Fire TV* . (tarih yok). https://play.google.com/store/apps/details?id=com.amazon.storm.lightning.client.aosp&hl=en_GB adresinden alındı
20. Dix, A., Finlay, J., Abowd, G., & Beale, R. (2004). *Human–Computer Interaction* (s. 318-364). içinde

Solidity Dili için Statik Analiz Aracı

Ahmet Gül ve Alper Şen

Boğaziçi Üniversitesi, Bilgisayar Mühendisliği Bölümü
{ahmet.gul1,alper.sen}@boun.edu.tr

Özet. Günümüzde blokzinciri teknolojisi çözümleri yaygınlaşmaktadır. Bu çözümleri uygulamak için, Ethereum Sanal Makinesi üzerinde çalışan bir programlama dili olan Solidity kullanılarak geliştirilen akıllı sözleşmelerin, yaygınlaştırılmadan önce potansiyel hatalarının tespit edilmesi önem kazanmıştır. Statik analiz kaynak kod üzerinden, geliştirme sırasında hataların tespit edilmesinde kullanılan bir yöntemdir. Biz bu çalışmada, akıllı sözleşmelerde yapılan temel kod yanlışlarını inceledik. Bunları kendi içlerinde gruplandırdık ve geliştiricilerin kodlarının kalitesini arttırabilmek için akıllı sözleşmelerdeki bu yaygın hataları tespit eden bir statik analiz aracı geliştirdik.

Anahtar Kelimeler: Ethereum, BlokZinciri, Solidity, Statik Analiz

Static Analysis Tool for Solidity Language

Ahmet Gül ve Alper Şen

Bogazici University, Department of Computer Engineering
{ahmet.gul1,alper.sen}@boun.edu.tr

Abstract. Nowadays, blockchain technology solutions are becoming widespread. To apply these solutions, it becomes important to identify potential errors in smart contracts which are developed by using Solidity, a programming language running on the Ethereum Virtual Machine, before these contracts are deployed. Static analysis is a method to detect errors on source code during the development. In this study, we examined the common errors on smart contracts. We grouped them and we developed a static analysis tool to detect these common errors on smart contracts for developers to increase their code quality.

Keywords: Ethereum, BlockChain, Solidity, Static Analysis

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

1 Giriş

Blokzinciri, varlık işlemlerinin dijital olarak imzalanmış, sıralı, kriptografik olarak bağlanmış bloklarını içeren, fikir birliği modeliyle yönetilen merkezi olmayan bir veritabanıdır [1]. Başka bir deyişle, blokzinciri herkesin tam yetkili olup kimsenin tek yetkili olmadığı, otorite bulunmayan bir yapıdır. Herhangi bir kişi ya da kuruma güven duymama blokzincirinin özünü oluşturmaktadır.

Merkezsiz bir yapının sonucu olarak, blokzinciri platformları genellikle kimsenin kontrol edemediği kripto para birimlerini oluşturmak için kullanılmaktadır. Örneğin; 2019 yılının haziran ayında en popüler kripto para birimlerinden olan Bitcoin 160.731.625.130 dolarlık bir market hacmine ulaştı [2].

Akıllı sözleşme, bir anlaşmanın şartlarını müzakare etmek için kurallarını saklayan, bu şartların otomatik olarak yerine getirildiğini doğrulayan ve daha sonra bu anlaşmayı gerçekleyen bir yazılım parçasıdır [3]. Blokzinciri teknolojisi güvenlik, dayanıklılık, sürerlik ve güvenilirlik özellikleri kapsamında değerlendirildiğinde, akıllı sözleşmelerin saklanması ve çalışması için en ideal ortamdır.

Ethereum, akıllı sözleşmelerin oluşturulması için özel olarak inşa edilmiş açık kaynaklı bir blokzinciri platformudur [4]. Akıllı sözleşmeler, Ethereum Sanal Makinesi üzerinde çalışan değiştirilemez uygulamalar olarak tanımlanabilir [5]. Değiştirilemezlik tarafların sözleşmelerin şartlarını ve kodlarını hatalı dahi olsa kabul ettiklerini, bunların herhangi bir tarafça değiştirilemeyeceği anlamına gelir.

Dijital varlıkları depolamaları ile meşhur blokzinciri platformları, üzerlerinde bulundurdukları akıllı sözleşmelerin açıkları ve hataları sebebiyle siber korsanları teşvik etmekte ve onlar tarafından sıklıkla hedef olarak seçilmektedirler. Bu sebepten dolayı, üzerinde herhangi bir hata barındırmayacak şekilde akıllı sözleşmeler geliştirmek önem arz etmektedir. Örneğin, Haziran 2016'da popüler bir akıllı sözleşme olan DAO'dan 150 milyon dolar çalındı [6]. Blokzinciri üzerinde akıllı sözleşme geliştirecek kişilerin hata tespiti yapabilecek araçlara ihtiyaç duyduğu açıktır.

Solidity, Ethereum Sanal Makinesi üzerinde akıllı sözleşmeler geliştirmek için oluşturulmuş bir programlama dilidir. Akıllı sözleşmeler Ethereum Sanal Makinesi üzerinde çalıştırılır ve bloklara dağıtılır. Akıllı sözleşmelerdeki açıklar, istenmeyen durumların önüne geçebilmek için statik ve dinamik analiz yöntemleri kullanılarak tespit edilebilir.

Statik analiz, bilgisayar programların çalıştırmadan kaynak kodlarını kullanarak analiz etme işlemidir [7]. Öte yandan, dinamik program analizi ise programları gerçek ya da sanal bir işlemcide çalıştırmak suretiyle gerçekleştirilen bir yazılım analiz yöntemidir [8].

Bu çalışmada Solidity dili ile yazılmış akıllı sözleşmelerdeki potansiyel açıkları tespit edebilmek için statik analiz yöntemini kullanan bir araç geliştirdik. Statik analiz yöntemini tercih etmedeki sebebimiz kodun tamamını kapsamayı garanti ettiği için bize olası tüm açık ve hataları gösterecek olmasıdır. Böylelikle aracımız ile yazılımcıların yazmış oldukları kodlardaki hataları, akıllı sözleşmeyi blokzincirine yaygınlaştırmadan önce tespit edebilmelerini hedefledik. Çalışmamızın ikinci bölümünde çalışmamızın daha iyi anlaşılabilmesi için Ethereum ve akıllı sözleşmeler ile Solidity hakkında genel bilgileri, daha sonra üçüncü bölü-

münde daha önce Ethereum ve Solidity'deki ataklar ve hataların tespiti hakkında yapılmış benzer çalışmaları, dördüncü bölümünde geliştirmiş olduğumuz aracın nasıl çalıştığını ve tespit edebildiği hataların neler olduğu hakkında bilgi vereceğiz. Daha sonra beşinci bölümde ise örnek bir akıllı sözleşmedeki bazı hataları göstermek için vaka incelemesi yapacağız. Altıncı ve yedinci bölümlerde ise yapmış olduğumuz deneyler ve gelecek çalışmalar hakkında bilgiler vereceğiz.

2 Temel Bilgiler

Blokzinciri verinin tek bir noktada saklanmadığı ve her paydaşta verinin tamamının bulunduğu bir yapıdır. İsteyen herkes paydaş olarak fikir birliği modeli doğrultusunda blokzincirine katılabilir. Blokzinciri bloklardan oluşur ve veri akışı tek yönlüdür. Blok içerisinde hash bilgisini, veriyi, işlemi ve zaman bilgisini bulunduran parçadır. Her bir blok eğer fikir birliği modeline göre uygunsa zincire eklenir ve bir daha değiştirilemez.

Ethereum bir bütün olarak ele alınacak olursa işlem tabanlı durum makinesi olarak görülebilir [9]. Bu platform üzerinde yapılan her bir operasyon için belirli bir gaz ücreti tanımlanmıştır. Bir başka deyişle, gaz ücreti operasyonun maliyetini belirtmektedir [10]. Gaz ücreti, Ethereum blokzincirinin hayatta kalmasındaki en önemli ögelerden biridir. Akıllı sözleşmeler madencilerin bilgisayarlarında çalışır ve burada yapılan her bir işlem için madenci o işlemin değeri kadar gaz kazanmış olur. Gaz ücreti, Ether cinsinden gerçek paraya çevrilebilmektedir.

Ethereum üzerinde iki farklı tür hesap bulunmaktadır [11]. Bunlardan birincisi, insanlar tarafından oluşturulabilen ve içinde Ether cinsinden paranın saklandığı harici hesaplar, diğeri ise; akıllı sözleşmeler tanımlanırken oluşturulan sözleşme hesaplarıdır. Mevcutta Ethereum blokzincirinde 1.7 milyon tane akıllı sözleşme yani diğer bir deyişle 1.7 milyon hesap bulunmaktadır ve bunlardan beş yüz bini aktif durumdadır [12]. Her bir hesabın kendine özel bir genel-anahtarı bulunmaktadır. Hesap adresleri genel-anahtarlar olarak değerlendirebilir. İstemciler akıllı sözleşmelerle bu adresler aracılığıyla haberleşebilir.

Ethereum Sanal Makinesi için geliştirilmiş birden fazla dil bulunmaktadır. Solidity nesneye dayalı bir programlama dili olup akıllı sözleşme geliştirebilmek için oluşturulmuş en popüler dildir. Solidity geliştirilirken C++, JAVA ve Phyton dillerinden esinlenilmiştir [13]. Bu özelliğinden dolayı pek çok yazılımcı için öğrenmesi kolay bir dildir.

Akıllı sözleşmeler bloklar üzerinde çalışmaktadır. Solidity de blok bilgilerine erişebilecek değişkenleri üzerinde tutmaktadır. Örneğin; blok hash numarası, sözleşme adres bilgisi, sözleşmeyi çağıran adres bilgisi, sözleşmeye gönderilen Ether miktarı, sözleşme hesabında bulunan Ether miktarı.

Akıllı sözleşmeler genel olarak birden çok fonksiyondan oluşmaktadır. Fonksiyonlar akıllı sözleşmelerin dışarıya açılan kapılarıdır. İstemciler ve akıllı sözleşmeler, ulaşmak istediği akıllı sözleşmeye üzerinde tanımlanan fonksiyonları çağırarak ulaşabilmektedirler. Solidity dili de akıllı sözleşmeler için fonksiyon geliştirmeyi desteklemektedir. Akıllı sözleşmelerdeki her bir fonksiyon çağırımı, bir işlem olarak değerlendirilmektedir.

3 İlgili Çalışmalar

Ethereum akıllı sözleşmelerindeki hata denetimi, aktif bir araştırma alanıdır. Wan ve diğerleri [14] block zinciri sistemlerindeki hata karakteristikleri hakkında bir çalışma yürüttüler. Bu çalışmalarında hataların blokzinciri sistemlerinin güvenilirliğine zarar verdiğini ve hataların birbirlerine benzer özelliklere sahip olduğunu ortaya koydular. Atzei ve diğerleri [15] Ethereum akıllı sözleşmelerine yönelik saldırıları içeren bir çalışma yürüttüler. Bu çalışma, akıllı sözleşmelere zarar verilebilen yaygın yanlışları, hataları ve bunların sebep olabileceği sonuçları göstermektedir.

Araştırmaların çoğunluğu, Ethereum Sanal Makinesi byte kodlarını analiz eden araçlar geliştirmeyi amaçlamıştır. Örneğin, Luu ve diğerleri [16], Ethereum Sanal Makinesi byte kodundaki açıkları kontrol etmek için sembolik yürütmeye dayanan OYENTE aracını geliştirdi. Bu araç önce kontrol akış çizgesini sonra da bir takım sembolik izler oluşturur. Bu işlemlerden sonra ise bir kısıt çözümleyici izleri analiz etmektedir. Mythril [17] de Ethereum Sanal Makinesi byte kodlarını analiz eden, sembolik yürütmeye dayanan bir araçtır. OYENTE'den farklı olarak, Mythril analiz yaparken konkolik testi [18] kullanır. Fontein ve diğerleri [19], Mythril'in OYENTE'den daha fazla hata tespit edebileceğini gösterdi. Bu araçlar, sözleşmelerin karmaşıklığının yüksek olması nedeniyle sözleşmelerdeki tüm yolları keşfetmeyi garanti edemeyip sadece bazı yolları keşfetmektedir.

Sembolik uygulama araçlarının dışında, akıllı sözleşmelerin güvenilirliğini arttırmak için geliştirilmiş statik analiz araçları da bulunmaktadır. Örneğin, Tsankov ve diğerleri [20] akıllı sözleşmelerin olası tüm davranışlarını keşfetmek için statik analiz aracı olan Securify'ı geliştirdi. Araç, güvenli olan ve güvenli olmayan davranışları sınıflandırma mantığına dayanmaktadır. Araç, sözleşmenin EVM bayt kodunu statik tek-atama formuna dönüştürür, ardından sözleşmelerle ilgili anlamsal gerçekleri ortaya çıkarır. Bu gerçeklere istinaden güvenlik kalıplarını kontrol ederek sözleşmelerin davranışlarının güvenli veya güvensiz olduğuna karar verir. Tikhomirov ve diğerleri [21] kaynak kodu ara bir gösterim formu olarak XML ayrıştırma ağacına çeviren ve bu ara gösterim formu üzerinde sorgulamalar yaparak güvenlik açıklarını tespit eden başka bir statik analiz aracı olan SmartCheck'i geliştirdiler.

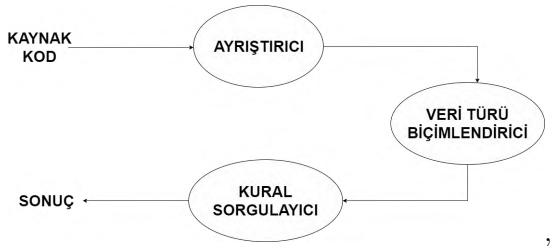
Güvenlik sorununa yol açmayan, ancak sözleşmelerdeki gaz tüketim maliyetini artıran modeller bulma konusunda da çalışmalar yapılmıştır. Chen ve diğerleri [22,23], akıllı sözleşmeleri ve Ethereum Sanal Makinesi derleyicisini optimize etmek için gaz maliyetli modeller bulmayı önerdi.

4 Yöntem

Bu bölümde aracımızın nasıl çalıştığını ve ne tür hatalar bulmayı hedeflediğimizi açıklamaktayız.

Geliştirmiş olduğumuz araç bir statik analiz aracıdır ve Solidity kaynak kodu üzerinden çalışmaktadır. Şekil 1'de aracımızın akış çizelgesi görülmektedir. Solidity dili ile yazılmış olan akıllı sözleşme kaynak kodunu girdi değeri olarak alır.

Daha sonra bu kaynak kodu, Solidity dili için geliştirilmiş Antlr Parser [24] aracılığı ile JSON ağacına yerleştirir. Araç bu yerleştirme işleminden sonra JSON ağacını XML ağacına dönüştürür ve tanımlamış olduğumuz 53 farklı kuralı [25] XPath [26] sorguları ile ağaç üzerinde tarar. Araç eğer bu kurallardan herhangi birini bulursa, kullanıcıya bunu gösterir.



Şekil 1. Program akış çizelgesi

Çalışmamız kapsamında 53 farklı kural tanımladık. Daha sonra bu kuralları kendi içlerinde dört farklı gruba ayırdık. Gruplama işlemi yaparken temel çıkış noktamız, tanımlanan bulgunun akıllı sözleşmeye ne oranda zarar verebileceğiydi. Kurallardan on bir tanesini 1. öncelikli yani akıllı sözleşmenin doğrudan saldırıya uğrayıp para kaybetmesine sebep olabilecek veya akıllı sözleşmede oluşturulan kuralların çalışmamasına sebebiyet verecek açıklar olarak sınıflandırdık. Daha sonra kalan kurallardan sekiz tanesini ise 2. öncelikli yani akıllı sözleşmenin doğrudan saldırı yemesine sebep olmayacak fakat akıllı sözleşmenin efektif çalışmamasına sebebiyet verecek kod parçacıkların tespitini içeren kurallar olarak sınıflandırdık. Kalan kurallardan yirmi altı tanesini ise 3. öncelikli olarak gruplandırdık. 3. öncelikli kurallar genel olarak kötü kokan kodları tespit etmektedir. Bu kodlar akıllı sözleşmeye doğrudan bir zarar yaratan veya efektik çalışmayan kodlar değildir. Bu gruptaki hatalar genel olarak yazılımcının eski versiyon fonksiyonlar ile iş yaptığını, güncel methodları kullanmadığı, kodun okunabilirliğini düşerecek ifadelerin kullanıldığını belirtir. En son kalan sekiz kuralı ise 4. öncelikli olarak değerlendirdik. 4. öncelikli kurallar Solidity dili için oluşturulmuş iyi yazılım örüntülerini içermektedir. Bu örüntülerin kullanılmaması yazılımcıya eksi bir değer katmamakla beraber; buradaki amacımız bu örüntüler tespit edilirse yazılımcıya doğru yolda olduğunu veya daha önceden onaylanmış ve bilinen bir çözüm kullandığını göstermektir.

Tablo 1'de, tanımlamış olduğumuz kuralların isimleri ve hangi öncelik kategorisine dahil edildiklerinin bilgisi bulunmaktadır.

5 Vaka İncelemesi

Çalışmamızın bu bölümünde aracımızın ne tarz hatalar tespit edebildiğini göstermek için Recover [27] isimli akıllı sözleşmeyi örnek olarak inceledik. Bu akıllı

Tablo 1. Kural isimleri ve öncelik seviyeleri

Kural İsmi	**Kural Tanımı**	**Öncelik Seviyesi**
Kural 1	Hatalı sıralanmış durum değiştirme sırası	1
Kural 2	Hatalı şekilde sayı bölme işlemi	1
Kural 3	Sayı taşma problemi	1
Kural 4	Payable fonksiyona karşılık payable fallback fonksiyonun bulunmaması	1
Kural 5	Zaman bilgisi kullanımı	1
Kural 6	Kilitli para durumu	1
Kural 7	Keccak ve Sha3 içerisinde tahmin edilebilir değer kullanımı	1
Kural 8	Hatalı fallback fonksiyon kullanımı	1
Kural 9	Kısıtlı yazım kuralı	1
Kural 10	Kısıtlı transfer kuralı	1
Kural 11	İş sıralama bağımlılığı	1
Kural 12	Boş döngü kullanımı	2
Kural 13	Koda gömülmüş adres değeri	2
Kural 14	Constant anahtar kelimeli fonksiyon kullanımı	2
Kural 15	Atomik olmayan iş tanımlanması	2
Kural 16	Dizi büyüklüğüne erişim	2
Kural 17	Storage ilişkili döngü	2
Kural 18	Sabit aralıklı döngü	2
Kural 19	Döngü tekrarı	2
Kural 20	Değişken veya fonksiyon görünürlüğünün private yapılması	3
Kural 21	Harici library kullanımı	3
Kural 22	Hatalı view fonksiyon tanımı	3
Kural 23	Sabit derleyici seçmeme problemi	3
Kural 24	Döngüde bayrak kullanımı	3
Kural 25	Send ifadesinin kullanımı	3
Kural 26	Transfer,send veya call ifadelerinin döngü içerisinde kullanımı	2
Kural 27	tx.origin kullanımı	3
Kural 28	throw kullanımı	3
Kural 29	suicide kullanımı	3
Kural 30	Hatalı string kontrolü	3
Kural 31	Boş if kullanımı	3
Kural 32	Hatalı şekilde dizi temizleme işlemi	3
Kural 33	sha3 ile hashleme problemi	3
Kural 34	Transfer ve TransferFrom fonksiyonlarının dönüş değeri	3
Kural 35	Değişken veya fonksiyon görünürlüğünün tanımlanmaması	3
Kural 36	approval fonksiyonu kullanımı	3
Kural 37	Belirsiz değişken türü kullanımı	3
Kural 38	Fonksiyon içerisinde revert kullanımı	3
Kural 39	Hatalı pure fonksiyon tanımı	3
Kural 40	Gereksiz return kullanımı	3
Kural 41	Gereksiz view fonksiyon kullanımı	3
Kural 42	assembly kullanımı	3
Kural 43	Çoklu kalıtım uyarısı	3
Kural 44	Gölge değer kullanımı	3
Kural 45	Gereksiz miras tespiti	3
Kural 46	Karşılıklı dışlama nesnesi (Mutex) paterni	4
Kural 47	İşlem sayacı paterni	4
Kural 48	Kontrol etkisi etkileşim paterni	4
Kural 49	Devre kesici patern	4
Kural 50	Bakiye limit kontrol paterni	4
Kural 51	Ölümcül patern	4
Kural 52	Hız kontrol paterni	4
Kural 53	Reddedici patern	4

sözleşme 774 satırdan oluşmaktadır. Fakat yalnızca bazı hataları inceleyeceğimiz için belirli kod parçacıklarını göstereceğiz.

Şekil 2'deki kod parçacağı Revocer akıllı sözleşmesinden alınmıştır. Bu kod parçacığında döngü içerisinde dizinin boyutuna .length ile erişim olduğu için fazladan gaz tüketimi olacaktır. Bunun sebebi döngünün her bir iterasyonunda .length kullanımından dolayı storage değişkenine yani bir sözleşme değişkenine erişim yapmasıdır. Daha sonra döngü içinde ise lokal değişkenler ile işlem yapılır. Bunun anlamı sözleşme değişkeni döngünün her iterasyonunda lokale kopyalanacak ve bu işlem için her seferinde fazladan gaz harcanmış olacaktır. Bu yüzden aracımız tarafından 2. öncelikli olan Kural 16 dizi büyüklüğüne erişim tespit edilmiştir.

```
contract Recover is IArbitrable {
 struct Claim {
    bytes32 itemID; // Relation one-to-one with the item.
    address finder; // Address of the item finder.
    uint finderFee; // Total fees paid by the finder.
    Status status; /* Status of the claim relative
    to a dispute.*/
    ...
 }
 Claim[] public claims; // Collection of the claims.
 ....
 function getClaimIDsByAddress(address _finder)
 public view returns (uint[] claimIDs) {
        uint count = 0;
        for (uint i = 0; i < claims.length; i++) {
            if (claims[i].finder == _finder)
                count++;
        }
        ...
    }
    ...
}
```

Şekil 2. Recover Akıllı Sözleşmesi'nden kod parçacığı

Burada fazladan gaz tüketimini engellemek için sözleşme değişkenindeki değer lokal bir değişkene atanmalı ve döngü bu lokal değişken üzerinden çalışmalıdır. Böylelikle sözleşme değişkeni lokale yalnızca bir kere alınacak ve fazladan gaz tüketiminin önüne geçilecektir. Şekil 3'te kod parçacığının düzeltilmiş hali bulunmaktadır.

```
 function getClaimIDsByAddress(address _finder)
 public view returns (uint[] claimIDs) {
        uint count = 0;
        uint localVariable = claims.length;
        for (uint i = 0; i < localVariable; i++) {
            if (claims[i].finder == _finder)
                count++;
```

```
        }
        ...
    }
```

Şekil 3. Recover Akıllı Sözleşmesi'nde düzeltilmiş kod parçacığı

Bu sözleşmede aracımız tarafından tespit edilen bir diğer durum ise 3. öncelikli olan Kural 40 gereksiz return kullanımı durumudur. Kod parçacığı üzerinden de görülebileceği üzere getClaimIDsByAddress fonksiyonunun üzerine returns ifadesi uint tipinde bir dizi döndürebilecek şekilde bir tanım yapılmıştır. Fakat Şekil 2'deki kod incelendiğinde herhangi bir değer döndürmediği görülmektedir. Yazılımcı burada ya returns ifadesini kaldırmalı ya da claims değişkenini return ile döndürmelidir. Yazılımcı eğer bu kullanımdan vazgeçmezse, akıllı sözleşmedeki bu fonksiyonu çağıracak bir sözleşme ya da istemci, kendi tarafında değer karşılayacak şekilde bir kod yazacak ve akıllı sözleşmeden değer dönmeyeceği için hataya düşmüş olacaktır.

Yine aynı akıllı sözleşmede bulunan Şekil 4'te bulunan farklı bir kod parçacağında ise 1. öncelikli olan Kural 2 hatalı şekilde sayı bölme işlemi durumu tespit edilmiştir. Burada bölme işlemi yapılırken SafeMath kütüphanesi kullanılmadığı için sayı taşma problemi yaşanabilir. Bununla beraber toplama işlemi sırasında da SafeMath kütüphanesi kullanılmadığı için sayı taşma problemi hatası yaşanma olasılığı fazladır. Her türlü matematiksel işlem yapılırken SafeMath kütüphanesini önermemizin sebebi Solidity'de tanımlanabilecek en büyük sayı ve en küçük sayı limitleri olan 2^{256} ve -2^{256}'nın kolaylıkla aşılabilmesinden dolayı akıllı sözleşmelerin hata alabilmesidir. SafeMath kütüphanesi ise bu tarz hataları kendi içinde yakalar ve programın güvenilirliğini tehlikeye atmaz.

```
if (_ruling == uint(RulingOptions.OwnerWins)) {
            item.owner.send(item.ownerFee +
            item.amountLocked);
        }else if (_ruling == uint(RulingOptions.FinderWins))
        {
            itemClaim.finder.send(itemClaim.finderFee +
            item.amountLocked);
        } else {
            uint split_amount = (item.ownerFee +
            item.amountLocked) / 2;
            item.owner.send(split_amount);
            itemClaim.finder.send(split_amount);
        }
```

Şekil 4. Taşma problemine sebebiyet verebilecek kod parçacığı

Örnek bir SafeMath [28] kütüphanesi kodu Şekil 5'te bulunmaktadır.

```
library SafeMath {

function mul(uint256 a, uint256 b) internal pure returns (
    uint256) {
   if (a == 0) { // Gaz optimizasyonu
```

```
      return 0;
    }
    uint256 c = a * b;
    require(c / a == b);

    return c;
  }

function div(uint256 a, uint256 b) internal pure returns (
    uint256) {
    require(b > 0);
    uint256 c = a / b;

    return c;
  }

function sub(uint256 a, uint256 b) internal pure returns (
    uint256) {
    require(b <= a);
    uint256 c = a - b;

    return c;
  }

function add(uint256 a, uint256 b) internal pure returns (
    uint256) {
    uint256 c = a + b;
    require(c >= a);

    return c;
  }
}
```

Şekil 5. Örnek bir SafeMath Kütüphanesi kodu

Şekil 2'deki kod parçacığında 3. öncelikli olan Kural 25 send fonksiyonun kullanımı sıkıntısı da tespit edilmiştir. Burada send yerine daha güvenli olan transfer fonksiyonu kullanılmalıdır. Bunu önermemizin sebebi transfer fonksiyonu eğer işlem hata alırsa throw ile hatayı fırlatmakta ve güvenli bir şekilde işlemi bitirmektedir. send fonksiyonu ise hata alırsa false dönmektedir. Bu durum send kullanıldığı zaman hatanın yazılımcı tarafından işlenmesi gereksinimi doğurur. Bununla beraber, yazılımcı hata işlemeyi sağlayacak bir geliştirme yapmazsa akıllı sözleşme hata alacaktır.

Şekil 6'da kod parçacığının düzeltilmiş hali bulunmaktadır.

```
if (_ruling == uint(RulingOptions.OwnerWins)) {
        uint256 totalAmount = item.ownerFee.add(
        item.amountLocked);
        item.owner.send(totalAmount);
    }else if (_ruling == uint(RulingOptions.FinderWins))
```

```
{
    uint256 totalAmount2 = itemClaim.finderFee.add(
                    item.amountLocked);
    itemClaim.finder.send(totalAmount2);
} else {
    uint split_amount = (item.ownerFee.add(
    item.amountLocked)).div(2);
    item.owner.send(split_amount);
    itemClaim.finder.send(split_amount);
}
```

Şekil 6. SafeMath Kütüphanesi kullanılarak düzeltilmiş bir kod parçacığı

Geliştirmiş olduğumuz araç kodların içeriği üzerinde herhangi bir değişiklik yapmamaktadır. Araç yalnızca bulguları sunmakta olup, kodları değiştirme işlemi yazılımcıdan beklenmektedir.

6 Deneyler

Etherscan [29] Ethereum blokzinciri üzerindeki blokların, işlem hareketlerinin, akıllı sözleşmelerin ve hesapların bilgisini görüntüleyebileceğiniz bir platformdur. Bu platform üzerinde Solidity kaynak kodu açık olarak paylaşılmış, Ethereum blokzinciri üzerinde çalışan, hesabında en çok Ether'e sahip 5 akıllı sözleşmeyi kendi aracımız ile inceledik. İnceleme sonucun tespit edilen hataların önceliği ve tespit edilen bulgu sayısı sözleşmelerin isimleri ile beraber Tablo 2'dedir.

Tablo 2. Akıllı sözleşme isimleri ve öncelik bazlı bulgu sayıları

Akıllı Sözleşme Adresi	**Öncelik 1**	**Öncelik 2**	**Öncelik 3**	**Öncelik 4**
WETH9	0	0	1	0
DigixDAO: MultiSigWallet	0	5	22	0
Golem: MultiSigWallet	0	7	32	0
MultiSigWalletWithDailyLimit	3	15	15	0
Wallet	10	22	25	0

Tablo 2'de görülebileceği üzere dünya üzerinde en yüksek mevduata sahip akıllı sözleşmelerde dahi efektif olmayan, kötü örnek oluşturulabilecek kodlar bulunmaktadır. Tabloda belirtilen akıllı sözleşmeler içerisinde en yüksek meblağa sahip olan WETH9 [30] sözleşmelesi hariç hepsinde efektif olmayan şekilde gaz tüketimine sebep olabilecek kodlar tespit edilmiştir. DigixDAO MultiSigWallet [31] ve Golem MultisigWallet [32] akıllı sözleşmeleri 1.öncelikte açık bulundurmamalarına rağmen yazılımcıları hataya sürükleyebilecek kötü kokan kodlar ve fazla gaz tüketimine sebep olabilecek kodlar barındırmaktadırlar. MultiSigWalletWithDailyLimit [33] ve Wallet [34] akıllı sözleşmeleri ise 1. öncelikli açıklar bulundurmaktadır. Tespit edilen 1. öncelikli açıklar; tahmin edilebilir değerlerden olan zaman bilgisine bağımlı kod yazımını anlatan Kural 5 zaman bilgisi

kullanımı ve akıllı sözleşmenin yalnızca Ether alıp, dışarıdaki herhangi bir hesaba aktaramayacağını ve bu sebepten sözleşmede bulunan Etherlerin içeride kilitli kalacağını anlatan Kural 6 kilitli para durumudur. İncelenen bu beş akıllı sözleşmede 4.öncelik olarak tanımladığımız herhangi bir iyi yazılım deseni tespit edilememiştir. Sözleşmeler bazı bulguları birden fazla kere barındırmaktadır.

7 Sonuçlar ve Gelecek Çalışmalar

Ethereum blokzinciri üzerinde geliştirilen akıllı sözleşmelerin sayısı her geçen gün artmaktadır. Bununla beraber akıllı sözleşmeler gündelik hayatımıza girdikçe Solidity dilini öğrenmesi gereken kişi sayısı da artacak ve bu dil ile ilgili araçlara ihtiyaçlar artacaktır. Geliştirmiş olduğumuz statik analiz aracı ile yazılımcılara kodlarında olabilecek potansiyel açıkları göstermeyi hedefledik. Daha ileriki çalışmalarda kullanıcı sayısının artması ve buna istinaden dilde olacak gelişmeler kapsamında yeni kuralların eklenmesi, statik analiz yöntemi ile bulunamayacak açıkların dinamik analiz yöntemleri ile yakalanması konusunda çalışmalar yürütülmesi hedeflenmiştir.

Kaynakça

1. Sultan, K., Ruhi, U., Lakhani, R.: Conceptualizing Blockchains: Characteristics Applications. In 11th IADIS International Conference on Information Systems (2018)
2. "CoinMarketCap" , https://coinmarketcap.com/, accessed: 2019-06-19
3. Migliorini, S.: Enhancing Blockchain Smart-Contracts with Proof-of-Location. In 10th International Conference on Geographic Information Science (2018)
4. Buterin, V.: Ethereum: a next generation smart contract and decentralized application platform. https://github.com/ethereum/wiki/wiki/White-Paper (2013)
5. Antonopoulos, A.M., Wood, G.: Masterin Ethereum. O'Reilly Media, 2018
6. "DAO Contract", https://bit.ly/2x8LNYB, accessed: 2018-10-04
7. J. Zheng, L. Williams, N. Nagappan, W. Snipes, J. P. Hudepohl, and M. A. Vouk, On the value of static analysis for fault detection in software. IEEE Trans. on Software Engineering, vol. 32, no. 4, pp. 240–253, 2006.
8. Ball, T.: The concept of dynamic analysis. In: Wang, J., Lemoine, M. (eds.) ESEC 1999 and ESEC-FSE 1999. LNCS, vol. 1687, p. 216. Springer, Heidelberg (1999)
9. Wood, G. Ethereum: A secure decentralised generalised transaction ledger, 2014.
10. Dannen, C. Introducing Ethereum and Solidity, 2017.
11. "Accounts", https://solidity.readthedocs.io/en/v0.5.7/introduction-to-smart-contracts.html#blockchain-basics, accessed: 2019-06-06
12. "Ecosystem Highlights", https://medium.com/@vikati/ranking-ethereum-smart-contracts-a27e6f622ac6, accessed: 2019-06-06
13. "Solidity", https://solidity.readthedocs.io/en/v0.5.7, accessed: 2019-06-06
14. Wan Z., Lo D., Xia X., Cai L. (2017) Bug Characteristics in Blockchain Systems: A Large-Scale Empirical Study. In IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)
15. Atzei, N., Bartoletti, M. And Cimoli, T.: A Survey of Attacks on Ethereum Smart Contracts SoK. In 6th International Conference on Principles of Security and Trust (2017)

16. Luu, L., Chu, D.H., Olickel, H., Saxena, P., Hobor, A.: Making smart contracts smarter. In: ACM SIGSAC Conference on Computer and Communications Security (2016)
17. Mueller, B. “Mythril - Reversing and bug hunting framework for the Ethereum blockchain.” [Online]. Available: https://github.com/bmueller/mythril/
18. K. Sen. Concolic testing. In Automated Software Engineering, pages 571–572, 2007.
19. Fontein, R.: “Comparison of static analysis tooling for smart contracts on the EVM”. (2018) Available from: https://telluur.com/utwente/Module
20. Tsankov, P., Dan, A., Cohen, D.D., Gervais, A., Buenzli, F., Vechev, M.: Securify: Practical Security Analysis of Smart Contracts. In ACM SIGSAC Conference on Computer and Communications Security (2018)
21. Tikhomirov, S., Voskresenskaya, E., Ivanitskiy I., Takheviv R., Marchenko E., Alexandrov, Y.: SmartCheck: Static Analysis of Ethereum Smart Contracts. In: IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB) (2018)
22. Chen T., Li X., Zhang X.: “Under-Optimized Smart Contracts Devour Your Money” In: IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER) (2017)
23. Chen T., Li Z., Zhao H., Chen J., Lio X., Li X., Zhang X.: Towards Saving Money in Using Ethereum Smart Contracts. In: ACM/IEEE 40th International Conference on Software Engineering: New Ideas and Emerging Results (2018)
24. "Solidity Antlr Parser" , https://github.com/federicobond/solidity-parser-antlr, accessed: 2018-07-01
25. "Kural Açıklamaları", https://github.com/ahmetgul93/Solidity-Static-Analyzer-Rule-Base, accessed: 2019-06-20
26. "XPath" , https://www.w3.org/TR/1999/REC-xpath-19991116/, accessed: 2019-06-06
27. "Recover", https://bit.ly/2x8qqqb, accessed: 2019-06-08
28. "SafeMath" , https://ethereumdev.io/safemath-protect-overflows/ , accessed: 2019-06-08
29. "Etherscan", https://etherscan.io/accounts, accessed: 2019-06-08
30. "WETH9 Contract", https://bit.ly/2KBPAqb, accessed: 2019-06-08
31. "DigixDAO: MultiSigWallet Contract", https://bit.ly/2RwDRK6, accessed: 2019-06-08
32. "Golem: MultiSigWallet Contract", https://bit.ly/2ht0kpz, accessed: 2019-06-08
33. "MultiSigWalletWithDailyLimit Contract", https://bit.ly/2xe7Tsw, accessed: 2019-06-08
34. "Wallet Contract", https://bit.ly/2RuA2Ff, accessed: 2019-06-08

Web Sitesi Saldırılarına Karşı Hızlı Çözüm Üreten Bir Uygulama

Murat Arslan[1], Burak Çarıkçı[1], and Murat Erten[2]

[1] İzmir Yüksek Teknoloji Enstitüsü
[2] Bakırçay Universitesi
{muratarslan,burakcarikci}@std.iyte.edu.tr;{murat.erten}@bakircay.edu.tr

Özet. Web siteleri saldırıya uğradığı zaman, saldırı ile ilgili istatistiki bilgi sağlamak, siteyi bu saldırılardan korumak kadar önemlidir. Bu çalışmada, saldırıya uğrayan web sitesinin bir klonunu oluşturan bir sistem geliştirilmiştir. Saldırganlar, Docker kullanılarak oluşturulan klon siteye yönlendirilmekte, öte yandan site çalışmaya devam etmektedir. Başarılı olduğunu düşünen saldırganın ise faaliyetleri kayıt edilerek istatistiki bilgi toplanmaktadır. Sistem geliştirilmiş ve test edilmiş, sonuçları da bu bildiride sunulmuştur.

Anahtar Kelimeler: Ağ güvenliği, uygulama güvenliği, siber saldırı istihbaratı.

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

An Application for Quick Response to Website Attacks

Murat Arslan[1], Burak Çarıkçı[1], and Murat Erten[2]

[1] Izmir Institute of Technology
[2] Bakırçay University
{muratarslan,burakcarikci}@std.iyte.edu.tr;{murat.erten}@bakircay.edu.tr

Abstract. When web pages undergo attacks, it is just as important to collect information and statistics regarding the attack as protecting the site. In this work a system was developed where a clone of the web page being attacked is created using docker and the attacks are diverted there. While the attacker thinks that he is performing an attack on the web page, statistical information is collected regarding his/her activities but the web page continues to function normally. The system was successfully implemented and tested, and the results are presented.

Keywords: network security · application security · cyber threat intelligence.

1 Introduction

In our changing and developing world, everything is getting connected with networks. Websites and application servers have the highest percentage in that space. Hence, web site security is becoming much more critical parallel with these improvements and, at the same time, hackers become equipped with the most capable tools and talents. Similar tools and approaches are required on the side of security domain to protect the sites and to gather as much intelligence on the attacks as possible.

Firewalls are one of the tools used to stop attackers from penetrating into servers and networks. Web Application Firewalls (WAF) do not deceive hackers but if an attacker is caught by WAF, payloads can not reach the web application because he is banned or that payload is banned. Hackers, on the other hand, detect the WAFs and apply developed techniques to bypass these [1].

Intrusion Detection/Prevention Systems (IDS/IPS) are also being used against attacks which exploit this vulnerability in servers and networks. They only offer protection against known attacks and are vulnerable to false positive or false negative detections. Anomaly based detection is also possible, but network performance is adversly affected when this approach is adopted [2, 3].

As mentioned before, it is important to protect web sites but gathering statistical and other information on these attacks is just as important. The above mentioned approaches do not provide any detailed information regarding the

attack or attackers. Honeypots, on the other hannd, are tools usually used to attract intruders and collect cyber intelligence. Vulnerable applications are installed on Honeypots with the purpose of attracting hackers. Honeypots, however, are standalone servers dedicated for this purpose, they are usually memory hungry and require considerable resources to maintain. They are also created once and need to be updated once compromised [4].

Our proposed approach involves introducing a clone of the web site under attack and divert the attackers to this clone. By doing so we do not only protect the web site but also gather intelligence. Using the clone website, the hackers are deceived and redirected to the docker container. The attacker continues to attack under the impression that he/she is successful. Even if the attackers find a way to bypass the docker, they need to change certain files for web server applications which causes extra work for them.

In the following paragraphs we shall describe the proposed system and present the results obtained through simulating cyber attacks to a web server.

2 System Description

This study offers a method to protect the elements connected to the Internet from web-side and network-side cyber-attacks. Our first aim is to detect attacking vectors to the website. Intrusion detection system (IDS) is used to detect these attacks. Special characters and log analysis can be used for this purpose. There are many IDS products for network analysis (Snort, OSSEC, Sagan, etc.). We have chosen to use Snort which offers the benefits presented in [5] . After detecting the attack, victim website is cloned with the help of virtualization technology. Cloned website is deployed and attacker is redirected to it. Meanwhile fake data/databases are produced and shown to the attacker. Attacker thinks that he/she is successful and keeps digging. Attacker's every move is logged and monitored by administrator of the server. The aim of this clone website is to protect the main website and analyse the behaviour of the attackers.

In our project we are using Docker as a virtualization technology. Python, PHP and Javascript are chosen for scripting languages.

To summarize the above actions; when an attacker tries to hack our web application by sending malicious payloads repeatedly, our IDS system (i.e. Snort) detects the first malicious payload and it produces an alert. These alerts are parsed by our engine (written in Python3). The engine then creates a docker container cloning the actual website. After deployment, the attacker is now redirected to clone docker container. He/she thinks that he/she is successful and keeps sending packets. Docker container is isolated from the real web server so while the attacker is hacking fake-web site, real web site continues to perform its operations. Every move/payload is collected by the system and stored to the database (which is hosted on Amazon Web Services (AWS) and running as Postgresql). Database is accessible to both the clone machines and our web application which is used for sharing intelligence with customers. Customers, in our case, are the subscribers who are using the described system to protect their

web sites. Database transfers the new entries to the web application and these entries are parsed and depicted in the front end.

As stated above, once the IDS system detects an attack it throws an alert and a docker container is deployed for this intruder only, and the .htaccess file is modified for redirection. After this point attacker is redirected to the cloned web page. For each attack a separate docker container is created and as they only occupy a small memory area, this approach does not create too much load for the server. Attacks are also prevented from affecting each other through this approach. As an example, if the intruder has an IP address as 123.1.2.3, then a container is activated for this particular IP and the logs of the attacks from this IP are kept in the database under this identity.

When a new attack entry is detected and this information is inserted into the database, the system sends it to the our web application. Web application parses it and shares it with the customer. An email notification is also sent to the customer. The system architecture is shown in Fig. 1

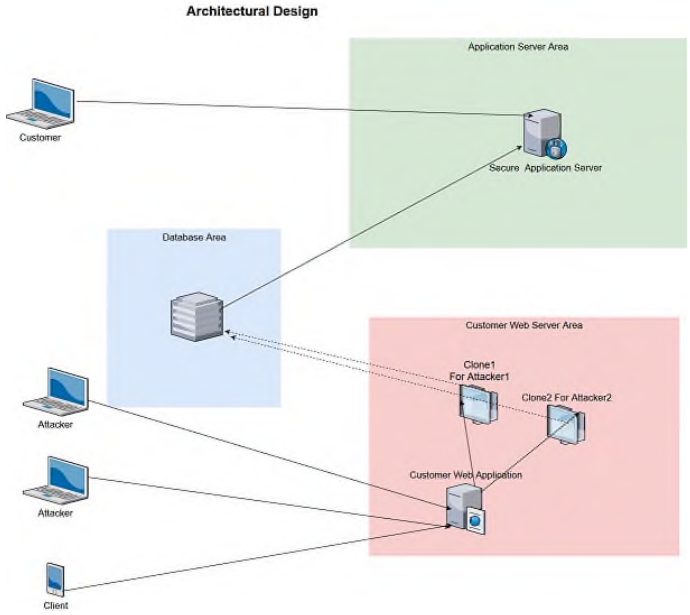


Fig. 1. The system architecture

3 Results

Testing the proposed architecture is performed using a victim machine which is hosted on AWS. All attacks are sent from our local machine. Victim machine can normally be found by automatic bots which scans all the internet.

Snort is configured to throw alerts to be processed further. If Cross-site Scripting (XSS) attack is detected (< script alert(1)> payload is sent), for example, Snort throws an alert, "XSS Detected", using the pre-defined rules. This alert consists of attacker's IP address, attacker's port number, affected parameters, victim's IP address, victim's port number, severity and activity of the attacker. This information is passed on to our core engine running on our server. In this study only four attack types, cross-site scripting (XSS), form based sql injection (SQLi), error based SQLi, and Internet control message protocol (ICMP) attacks were implemented.

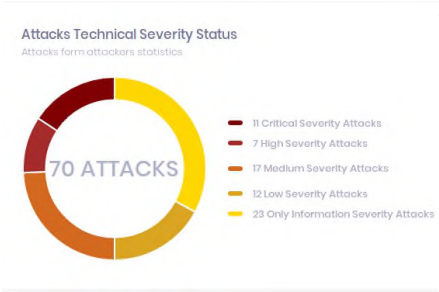


Fig. 2. Presentation of attack severity

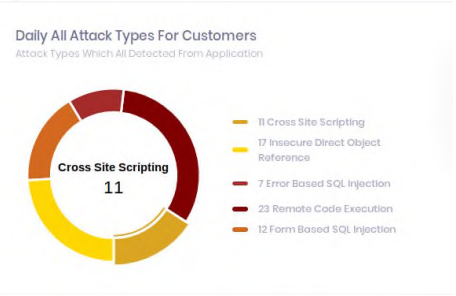


Fig. 3. The attack types

Alert is then parsed by the core engine. Core engine is continuously communicating with the database server where there is a table in the database for storing parsed alerts. Database stores every attack in detail.

Database communicates with our web application which is used for sharing cyber threat intelligence. All data are sent to the web application. Python/Django is using ORM (Object Relational Model) query models to query the database. If it detects a new entry, this log is parsed and shown in the user interface to the particular client who has been attacked. The displays depicting the attack severity and attack types, as presented to the clients, are shown in Fig. 2 and Fig. 3.

Following are the list of actions performed:

- Payloads are logged.
- Core engine parses log
- Logs are written in database
- Backend sends query to database
- Informations are shown on the user interface in less than 1 second, thanks to the efficient infrastructure of AWS and Django.

The logs kept by the system is shown in Fig. 4 and the intelligence information provided to the clients are shown in Fig. 5.

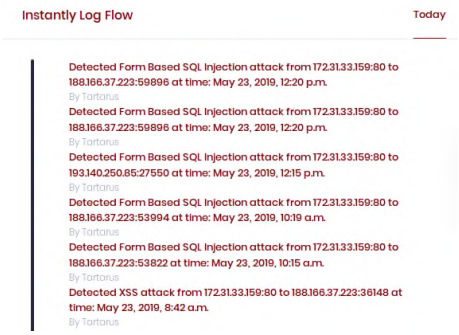


Fig. 4. The log file of attacks



Fig. 5. Cyber intelligence data

4 Discussions

The results presented demonstrate the data obtained using the proposed system. The system works successfully both for protecting the web site and for collecting the necessary intelligence on the attacks performed. The IT managers of the respective clients would be able to access our server and display the types of attacks they are being subjected to and from where, hence they may be able to take precautions accordingly.

An alternative approach would be a different architecture where the product is running on a dedicated server. In this case, user traffic will first pass from this server and then routed to the user website. Container can now be deployed on product server as soon as an attack is detected. Through this approach the user web site will be protected from Denial of Service (DoS) attacks and its normal operation will not be affected due to bandwidth being exhausted. In this case, however, the static web pages will have to be kept at our server and shown to the attacker during the cloning process. As the application will be hosted on a service such as AWS, there will not be bandwidth or space restrictions but the cost will obviously increase. The alternative architecture is shown in Fig. 6

One other problem with the current approach arises when the intruder performs an attack to an online shopping site, for example, and starts shopping, and starts performing the attack after putting a few items into the cart. In this case, because the clone would be showing the static pages, he/she would not be able to see the cart hence will be able to detect that he/she has been compro-

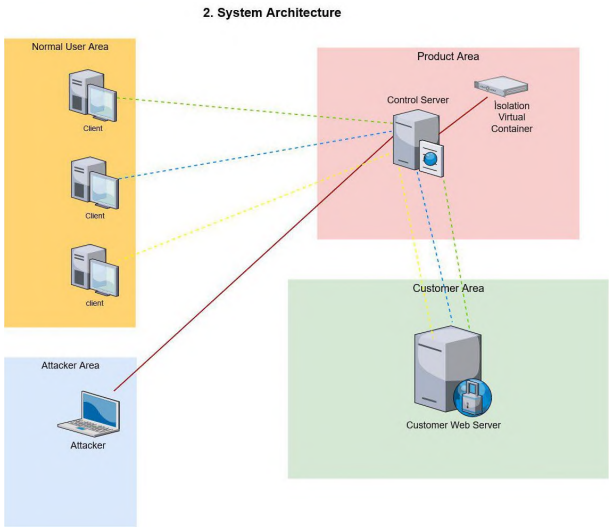


Fig. 6. Alternative architecture

mised. Experimental work should also be carried out to measure the overheads introduced by this system, and the data to be gathered should also be decided, may be case by case. We intend to study these issues as well as the application to new attack types as future work.

References

1. Clincy, V., Shahriar,H., "Web Application Firewall: Network Security Models and Configuration", In:2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC),pp. 835-836, Tokyo, (2018)
2. Khan, S., Motwani, D., "Implementation of IDS for web application attack using evolutionary algorithm," In:2017 International Conference on Intelligent Computing and Control (I2C2),pp. 1-5, Coimbatore, (2017)
3. Gaddam, R., Nandhini, M., "An analysis of various snort based techniques to detect and prevent intrusions in networks proposal with code refactoring snort tool in Kali Linux environment," 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT), pp. 10-15, Coimbatore, 2017
4. Catakoglu,O.,Balduzzi, M., Balzarotti, D., "Attacks Landscape in the Dark Side of the Web", In:Proceedings of the 32nd Annual ACM Symposium on Applied Computing (SAC), ACM, Marrakech, Morocco (2017)
5. CyberPersons homepage https://cyberpersons.com/2017/07/03/how-to-install-snort-and-use-as-web-application-firewall/. Last accessed 27 May 2019

Dijital Enerji IOT Uygulaması: KIO

Gökhan Akyol[1], Okan Akbay[2] ve Serkan Kaba[3]

[1,2,3] Klemsan Elektrik ve Elektronik A.Ş. Kemalpaşa, İZMİR
gokhanakyol@klemsan.com.tr, okanakbay@klemsan.com.tr,
serkankaba@klemsan.com.tr

Özet. Enerji yönetimi, çevresel ve ekonomik hedefleri göz önünde bulundurarak ihtiyaçların karşılanması için tedarik, dönüşüm, dağıtım ve ernerji kullanımının proaktif, organize ve sistematik koordinasyonudur. Enerji izleme ve yönetimi yazılımı, enerjinin verimli kullanılmasını sağlar. Klemsan Elektrik ve Elektronik tarafından geliştirilen, web tabanlı bir enerji IOT (Internet of Things) uygulaması olan KIO (Klemsan Internet Objects), firmaların enerji verimliliği ve enerji tasarrufu sağlamalarına çözüm üretir. KIO genişleme ve performans için modüler ve ölçeklenebilir bir mimari barındırır. KIO, müşterilerin ihtiyaçları doğrultusunda özelleştirilebilir bir yapıya sahiptir. Bu çalışmada, uygulamaya genel bakış ve uygulama özellikleri, uygulama alanları ve uygulamanın gelecekteki özellikleri gösterilmektedir.

Anahtar Kelimeler: Enerji Yönetimi Yazılımı, Nesnelerin İnterneti, Enerji Verimliliği.

Digital Energy IOT Application: KIO

Gokhan Akyol[1], Okan Akbay[2] ve Serkan Kaba[3]

[1,2,3] Klemsan Electric and Electronics, Kemalpasa, IZMIR
gokhanakyol@klemsan.com.tr, okanakbay@klemsan.com.tr,
serkankaba@klemsan.com.tr

Abstract. Energy management is the proactive, organized and systematic coordination of procurement, conversion, distribution and use of energy to meet the requirements, taking into account environmental and economic objectives. Energy monitoring and management software provides used of energy efficiently. KIO (Klemsan Internet Objects) is a web-based energy IOT (Internet of Things) developed by Klemsan Electric and Electronics, enables firms to supply energy efficiency and energy savings. KIO has modular and scalable architecture for expanding and performance. There is a customizable structure for operational needs

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

in KIO. This paper shows the application overview and properties, application areas and future work of this project.

Keywords: Energy Management Software, Internet of Things, Energy Efficiency

1 GİRİŞ

Enerji yönetimi kaliteden, güvenilirlikten veya çevre koşullarından ödün vermeden ve üretimi azaltmadan, enerjinin verimli bir şekilde kullanılma yönünde yapılandırılmış ve organize edilen disiplinli bir çalışmadır. Enerji yönetiminin temel prensibi, enerji verimliliğinin arttırılması, ekonomik kalkınma ve sosyal refahı engellemeden, mevcut enerji kayıplarının kalite ve performasnın düşürülmesine engel olmaksızın tüketilen enerji miktarını en aza indirmektir [1]. Aşağıda enerji yönetimi yazılımının birçok avantajını görmekteyiz.

- Planlanan bakımlarla verimli çalışma sağlar.
- Arızalara anında müdahale ile duruş sürelerini minimum indirir.
- Enerji şebekesini analiz ederek enerji kalitesi kaynaklı makina arızalarını önler.
- Analiz ve raporlamalar otomatik olarak yapılır, böylece insan kaynaklı hatalar ortadan kaldırılırken zaman ve iş gücünden tasarruf sağlanır.
- Geçmiş verilere kolaylıkla erişilerek farklı tarihlerdeki parametreler karşılaştırılabilir.
- Uzaktan izleme ve alarm yönetimi ile gereksiz servis gönderimlerinin önüne geçilir.

IOT tabanlı enerji yönetimi yazılımı, enerji yönetimi ve enerji yönetim sistemi sertifikasyonu (ISO 50001) için önemli bir bileşendir. IOT tabanlı enerji yönetimi yazılımı, enerji kullanımı ve bununla bağlantılı maliyeti ve karbon salınımı azaltır [2]. Literatürde sadece web tabanlı enerji IOT yazılımının yanı sıra SCADA tabanlı enerji yönetimi yazılımı da mevcuttur. SCADA tabanlı enerji yönetimi yazılımı üretim makinalarından tüketim verilerini toplar, verileri merkezi yönetim sistemine aktarır ve istenmeyen durumlar oluştuğunda alarm gönderir [3]. Ayrıca, enerji ve sensor uygulamalarını barındıran IOT platformları yer almaktadır. Schneider Electric firmasının EcoStruxure [4], Siemens firmasının MindSphere [5], PTC firmasının Thingworx [6] IOT platformları müşterilere geniş bir ekosistem sağlamaktaıdır. KIO web tabanlı bir enerji ve sensor IOT uygulaması olup SCADA sistemine benzer bir uygulamayı içinde barındırarak SCADA yazılımlarının bir adım önüne geçmektedir.

2 UYGULAMAYA GENEL BAKIŞ VE UYGULAMA ÖZELLİKLERİ

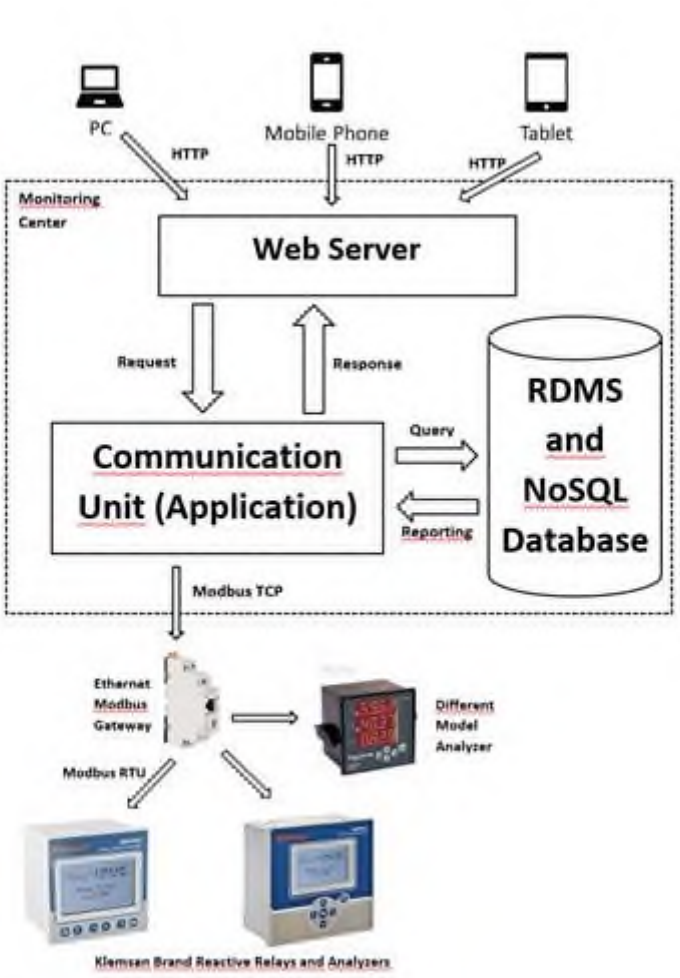
Klemsan Elektrik ve Elektronik olarak enerji analizörleri, reaktif güç kontrol röleleri, multimetre cihazları, sensörler, erişim noktaları ETOR (Ethernet to RS485), WTOR (Wi-fi to RS485) ve GTOR (GPRS to RS485) üretmekteyiz. Elektrik ve elektronik

sektöründe 40 yıllık bilgi ve tecrübeye sahip olan Klemsan Elektrik ve Elektronik, enerji izleme ve yönetimi yazılımı çıkarma mecburiyetindedir. Bu bağlamda, kurum içi girişimcilik ile doğan ve tamamen yerli yazılım ürünü olan KIO, Ocak 2017 itibariyle ilk versiyonuyla yayımlanmıştır. KIO açık modüler yapısıyla ölçeklenebilir uygulamalar ve sektöre özel çözümler sunar. KIO enerji yönetimi alanındaki anlık izleme, raporlama, faturalama, alarm yönetimi ve enerji verimliliği ekranları gibi tüm gereksinimleri tek bir çerçevede toplayarak müşterilere paket çözümler üretmektedir. Klemsan cihazları seri haberleşmeyi (MODBUS RTU) desteklerken KIO, TCP / IP katmanında (MODBUS TCP) haberleşmektedir. Dolayısıyla KIO ile cihazlar arasında protokol dönüştürücüler koyularak haberleşme sağlanmaktadır. KIO, protokol dönüştürücüler sayesinde uç noktalara (cihazlara) belirli zaman aralıklarında sorgu göndermektedir. Sorgunun cevabı başarılı ise haberleşme sağlanır ve uç noktaların verileri KIO'da görülmeye başlar. Sorgunun cevabı başarısız ise KIO belirlenen tekrar deneme sayısı kadar sorgulamaya devam eder. Eğer sorgu hala başarısız ise cihazın bağlantıları ya da protokol dönüştürücüler kontrol edilmelidir. KIO, protokol dönüştürücü olmadan direkt Ethernet üzerinden de haberleşme sağlamaktadır. Uygulamanın genel topolojisi Şekil 1 üzerinde görülmektedir.



Şekil 1. KIO Uygulama Topolojisi

Şekil 1'de KIO yazılımı merkezi haberleşmeyi başlatan yazılımdır. KIO, sunucu-istemci mimarisinde istemci rolünde çalışır. Sahadaki cihazlar ise sunucu rolündedir. KIO, sahadaki cihazlara bağlanan protokol dönüştürücülerin (ETOR, WTOR, GTOR) IP ve port bilgilerini alarak sorgulama başlatır. Haberleşmesi sağlanan cihazlar, kullanıcı yetkilerine göre web ara yüzü üzerinden takip edilmektedir. Uygulamanın sistem mimarisi Şekil 2'deki gibidir.



Şekil 2. KIO Sistem Mimarisi

Şekil 2'de görüldüğü üzere KIO sistem mimarisinde üç adet yapı yer almaktadır. Web sunucu, haberleşme birimi (yazılımı) ve veritabanı sunucusundan oluşan sistem mimarisinde web sunucu HTTP isteğini web tarayıcı üzerinden almaktadır. Haberleşme birimi, bu isteği işleyerek cihazlardan veri okuma, alarm üretme, raporlama ve veri alışverişi gibi görevleri üstlenir. Veritabanı sunucusu ise haberleşme biriminden gelen raporlama sorgularını işleyerek kullancılara tarayıcı üzerinden sunmaktadır.

KIO uygulamasının alt yapı ve mimari özellikleri aşağıdaki gibidir.

Web Tabanlı Sistem: Kullanıcılar herhangi bir platforma bağlı kalmadan standart bir web tarayıcı üzerinden sisteme erişim sağlayabilirler.
Genişletilebilir Modüler Yapı: Genişletilebilir modüler yapı ile kullanıcıların ihtiyaçları doğrultusunda çeşitli ekranlar özelleştirilebilmektedir.
Marka Bağımsız Entegrasyon: Klemsan veya farklı markaların ürünleri sisteme eklenebilir ve ürünlerin enerji tüketimleri sistem üzerinden takip edilebilmektedir.
Gelişmiş Veritabanı: Gelişmiş veritabanı özellikleri ile cihaz ve parametrelerin detaylı takibi ve kayıtların tutulması sağlanır.
Zaman Dilimi Desteği: Zaman dilimi desteği ile farklı bölgelerde bulunan cihazlar için farklı zaman dilimi tanımlamaları yapılabilir.
Güvenlik: HTTPS desteği, kullanıcı bazlı şifreleme, işlev ve profil bazlı kullanıcı yetkilendirmesi ile üst seviyede sistem güvenliği sağlanır.
Yük Paylaşımı: Yük paylaşımı ile sınırsız sayıda cihazla haberleşme sağlanabilir. Tek sunucu ile 2000 sayıda cihaz desteklenmektedir.

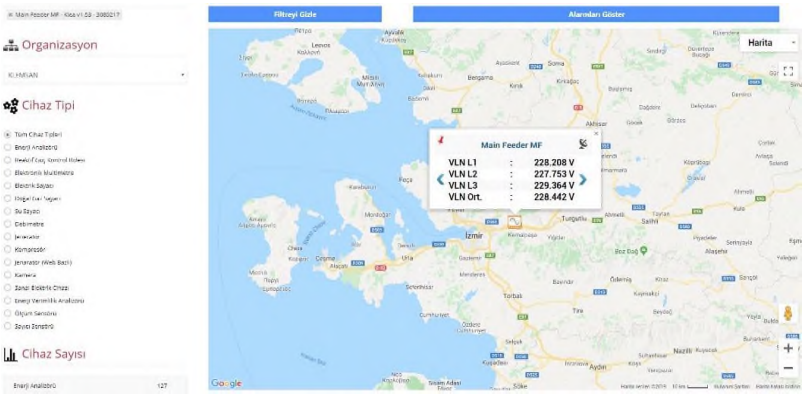
KIO'nun fonksiyonel özelliklerini listelemek gerekirse;

Gerçek Zamanlı İzleme: Gerçek zamanlı izleme özellikleri ile sistemdeki tüm cihazların anlık verileri, cihazların durumları ve alarmlar takip edilebilmektedir.
Gelişmiş Raporlama: Rapor formatı oluşturma, otomatik raporlama, hızlı faturalama özellikleri ile alarmlar raporlanabilmektedir. Şekil 3, birden fazla cihaza ait birden fazla parametrenin raporlanabildiği ekrandır.



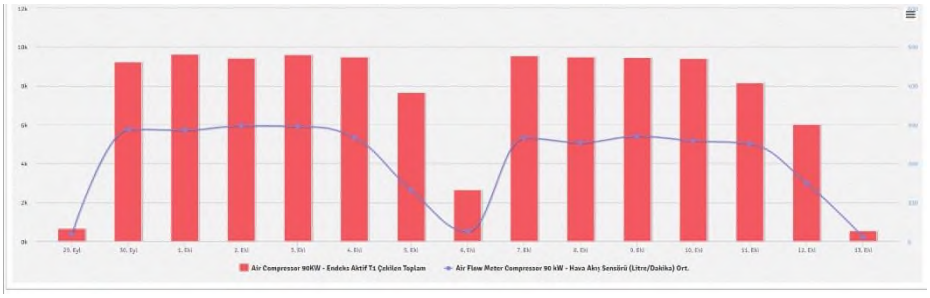
Şekil 3. KIO Gelişmiş Raporlama

Harita Modulü: Farklı bölgelerdeki tesis veya şubelerdeki cihazlar harita üzerinden işaretlenerek cihazlar harita üzerinden takip edilebilmektedir. Şekil 4'te KIO harita izleme modulü yer almaktadır. Harita ekranında işaretlenen cihazlar, bu ekran üzerinden izlenebilmektedir. Çeşitli bölgelerde işaretlenen cihazların üzerine gelindiğinde cihazın anlık verileri görülebilmekle beraber cihazın, işletmenin hangi noktasında olduğu bilgisi de paylaşılmaktadır.



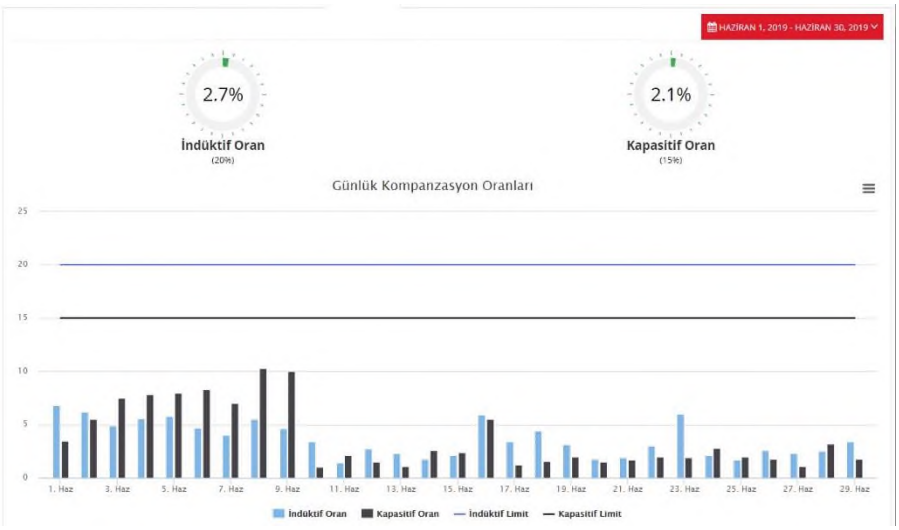
Şekil 4. KIO Harita Modulü

Dashboard, Grafik İzleme ve Tüketim Karşılaştırma Ekranları: Kullanıcılar isteklerine göre uygulamanın sağladığı özet ekranları (dashboard) ekleyebilir, grafiksel bazlı izleme ekranı ile farklı ölçüm parametreleri karşılaştırılabilir, tüketim karşılaştırma ekranı ile aynı tip cihazların enerji tüketimleri tek bir ekran üzerinden görülebilmektedir. Şekil 5, tüketim verisi (sütun) ile hava akışının (çizgi) ilişkisi gösterilmektedir.



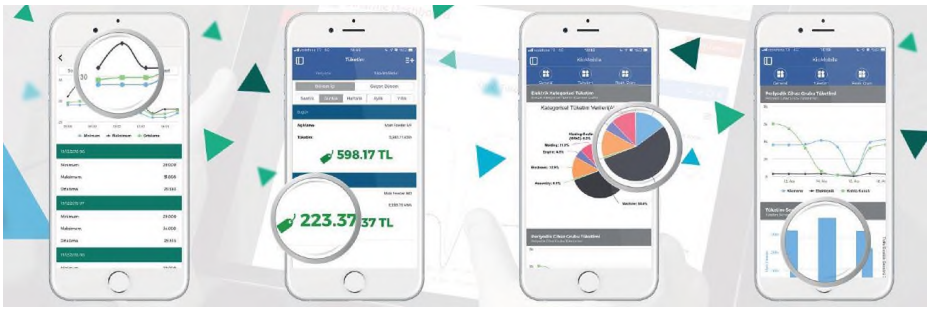
Şekil 5. KIO Grafiksel Raporlama

Kompanzasyon Takibi: Elektrik tüketimleri kontrol edilerek reaktif ceza riski minimuma indirilir. Şekil 6'da KIO'da kompanzasyon takibinin yapıldığı ekran görülmektedir. Belirlenen indüktif ve kapasitif oranlara göre reaktif cezaya yaklaşmadan sistem alarm bildiriminde bulunmaktadır.



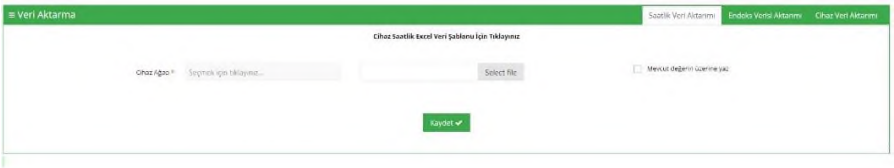
Şekil 6. KIO Kompanzasyon Takibi

Alarm Yönetimi: Alarm yönetim sistemi ile sistemde tanımlı cihazlar için alarm parametreleri oluşturulabilir, tarih ve zaman bazlı alarm kayıtları tutulabilir ve yine sistem üzerinden alarm durumları takip edilerek gerektiğinde sonlandırılmaktadır.
Fatura Yönetimi: Parametrik yapıda oluşturulan fatura tarifelerine göre fatura oluşturulabilir, cihazların dönemlere ait fatura tutarları görülebilir, tahsilatı yapılmış faturalar sistem üzerinden takip edilebilmektedir. Ödemesi yapılmamış faturalar için cihazların enerji kesme işlemi fatura yönetimi ile sağlanmaktadır.
Tek Hat Şeması (Mimik Diyagram): İşletmelerin tek hat şemaları sistem üzerinden tasarlanabilir. SCADA sisteminin benzer özellikleri web tabanlı olarak paylaşılmaktadır. Verilerin anlık olarak takip edildiği, uygulama üzerinden cihazlara komut gönderilebildiği, cihazların anlık, periyodik ve arşiv grafiklerinin çizilebildiği mimik ekranları tipik SCADA sisteminin özelliklerini taşımaktadır.
Enerji Verimliliği: Enerji verimliliği ekranları ile işletmelerde yer alan üretim süreçleri optimize edilebilir, birim maliyet hesabı yapılmaktadır. Karbon salınımı, TEP (Ton Eşdeğer Petrol) hesabı ve ağaç sayısı ile çevresel enerji yönetimini de olanak sağlanmaktadır.
Mobil Uygulama: Akıllı telefon ya da tablete indirilen mobil uygulama sayesinde işletmelerin enerji yönetimi yapılabilmektedir. Mobil uygulamanın, Android ve IOS sürümleri mevcuttur. Şekil 7'de KIO mobil uygulamasının ekran görüntüleri yer almaktadır.



Şekil 7. KIO Mobil Uygulaması

Çok Koşullu Durum: Sistemde yer alan analog, digital ve türetilmiş değerler üzerinden mantıksal işlemler gerçekleştirilerek çeşitli kontroller (vana açma, enerji kesme) sağlanmaktadır.
Maliyet Analizi: Muhasebe ve finans birimleri için destekleyici veriler sağlamak amacıyla çeşitli enerji kaynaklarına göre bütçe hesaplamaları yapılabilir. Hedef ve gerçek maliyetler karşılaştırılabilir.
Veri Transferi: Excel veya benzeri dosya formatlarında ölçülen değerler ve parametreler sisteme aktarılır ve izlenmektedir. Şekil 8'de veri aktarma modülü ekran görüntüsü yer almaktadır. Saatlik ve endeks verilerinin yanı sıra cihazların da veri aktarımı sağlanmaktadır.



Şekil 8. KIO Veri Aktarma Modulü

Entegrasyon: Sistem diğer yazılımlarla entegre halde çalışabilmektedir. Web servisler sayesinde muhasebe yazılımları ile fatura kesme konusunda ERP yazılımları ile de enerji yönetim modülü kapsamında veri alışverişi sağlanmaktadır.

3 UYGULAMA ALANLARI

KIO, enerji üretimi, enerji iletimi, yenilenebilir enerji, üretim tesisleri, bankalar, hastaneler, turizm tesisleri, alışveriş merkezleri ve ofis binaları gibi birçok farklı sektör için özelleştirilmiş çözümler sunmaktadır.

KIO bankalar için şubelerin anlık izlenmesinin ve kesintisiz enerji ihtiyacının garanti altına alınmasını sağlar [7]. KIO, bankacılık modülünde iklimlendirme, priz ve aydınlatmaların tüketim analizi, reaktif cezayı önlemek için kompanzasyon takibi ve şube

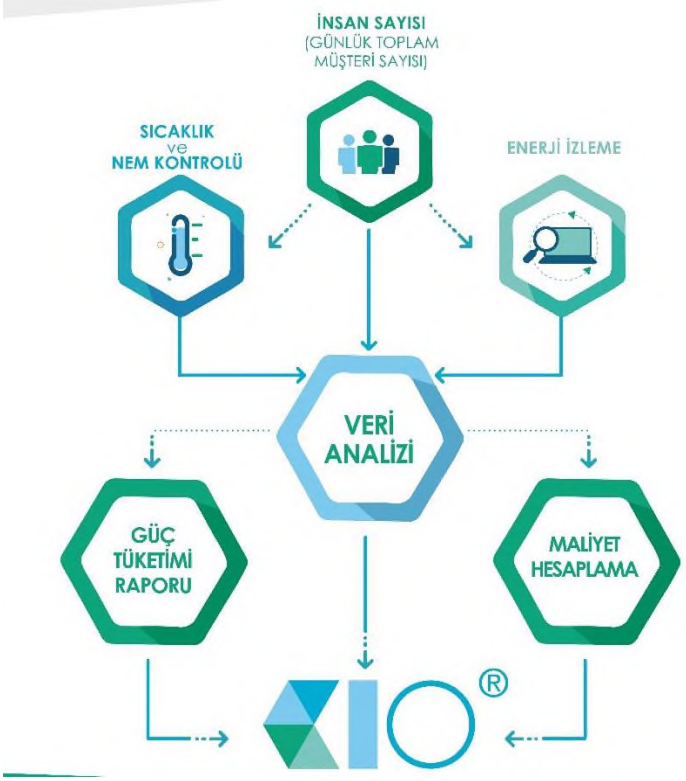
ekipmanlarının izlenmesi için jeneratör, HVAC ve sunucu sistemleri takibi yapılmaktadır. Şekil 9'da KIO'nun bankalar için uygulanacak çözümü görülmektedir. Bankalarda izlenen üç temel faktör yer almaktadır. Bunlar tüketim analizi, kompanzasyon takibi ve şube donanım takibidir.



Şekil 9. KIO Bankacılık Modülü

KIO'nun bir diğer uygulama alanı turizm tesisleridir. Otellerin enerji ve operasyonel verimliliğini arttırmak için KIO kullanılmaktadır. Otellerde manual yapılan enerji ölçümleri KIO sayesinde otomatik hale gelmiştir ve operasyonel verimlilik artmıştır. Ayrıca kompanzasyon takibi yapılarak reaktif enerjinin kullanım aşımları kontrol altına alınmaktadır. Elektrik dağıtım şirketinden gelen elektrik faturalarının karşılaştırılması KIO ile sağlanmaktadır. Buna ek olarak bütçe yönetim raporlarıyla elektrik, su, LNG, motorin gibi enerji kaynakları, otelde kalan aylık kişi sayısına göre oranlanarak maliyet analizi de yapılmaktadır.

KIO, alışveriş merkezi, zincir restoran, mağazalar ve perakende sektöründe de kullanılmaktadır. KIO alışveriş merkezi çözümünde enerji izlemenin yanı sıra fatura kesme en önemli özellik olarak kullanılmaktadır. Şekil 10, KIO'nun zincir restoranda uygulama yöntemi gösterilmiştir. IOT uygulamasına örnek olarak zincir restoranlarda insan sayımı, sıcaklık ve nem kontrolü, enerji izleme temel faktörlerdir. Performans ölçüm kriterleri raporlanarak birim maliyet hesabı yapılmaktadır.



Şekil 10. KIO Zincir Mağaza Modulü

KIO'nun bir diğer uygulama alanı hastanelerdir. Hastanelerde yer alan SCADA programı OPC sunucu yazılımıyla KIO'ya entegre edilerek veriler raporlanmaktadır. Ayrıca sanal cihaz tanımlamaları sayesinde kat ve bölüm bazlı enerji tüketimleri raporlanarak veri analizi yapılmaktadır. KIO hastane modülünde elektrik, su ve doğal gaz sayaçlarının tüketimleri de periyodik olarak ölçülmektedir. Ayrıca bu enerji kaynakları bütçe yönetim raporları ile hedef ve gerçek tüketimler karşılaştırılarak ay sonu gidişat tüketimleri kontrol edilmektedir.

KIO'nun en önemli uygulama alanlarından biri ise üretim tesisi ve fabrikalardır. Üretim tesisi ve fabrikaların en önemli gider kalemlerinden biri enerjidir. Enerjinin yönetimi ve tasarrufu üretim tesisi ve fabrikalar için önem arz etmektedir. Üretim tesislerinin, tükettiği enerjiyi etkin kullanabilmesi ancak ve ancak dijitalleşme ve proses optimizasyonu ile mümkündür [8].

Üretim tesisleri ve fabrikalar için pano ve trafo odalarının sıcaklık değerlerinin kontrolü çok önemlidir. Sıcaklık değerlerinin aşırı yükselmesi pano ve trafolara zarar verebilmektedir. Bu durumun önüne geçebilmek için sıcaklık sensörlerinin uzaktan izlenmesi ve anormal bir durumda yetkililere bildirimde bulunulması hem bakım planlamasına hem de olası zararı minimuma indirmeye fayda sağlayacaktır.

Üretim tesisi ve fabrikalarda izlenmesi gereken bir diğer etmen havadır. Makinalara monte edilen hava akış sensörleri sayesinde makinanın enerji tüketimi ile hava akışı arasında bir ilişki olduğu enerji yönetimi yazılımı ile analiz edilebilmektedir. Kompresörün enerji tüketimi ile hava tüketimi karşılaştırılarak fabrikada hava kaçağının olup olmadığı anlaşılmaktadır.

Tüketim değerleri makinalara yerleştirilen enerji analizörleri sayesinde hesaplanır. KIO, saatlik, günlük, aylık ve yıllık olarak tüketim değerlerini hesaplayarak raporlamaktadır. Aynı makinaların tüketim değerleri tüketim karşılaştırma ekranlarına gösterilir. Böylece hangi makinanın ne kadar enerji tükettiği parasal olarak da gösterilmektedir. KIO, üretimdeki makinaların üretim sayısını enerji tüketimi ile oranlayarak birim maliyeti analizi yaparak enerji verimliliğine katkı sağlamaktadır.

KIO modüler yapısı sayesinde her türlü sektöre paket çözümler sunarken aynı zamanda özelleştirilebilir yapısı sayesinde müşteri isteklerine cevap verir niteliktedir.

4 SONUÇ VE GELECEKTEKİ ÇALIŞMALAR

Çeşitli sektörlerde kullanılan KIO, alarm yönetimi sayesinde önleyici bakım hizmetlerine yardımcı olmaktadır. Farklı kategorilerin (makina, aydınlatma, iklimlendirme) enerji tüketimlerini göstererek hangi alanda enerji tasarrufu yapılması gerektiği ile ilgili bilgi vermektedir. Mimik ekranlar sayesinde uzaktan cihazların kontrolü sağlanırken cihazların anlık verileri canlı olarak izlenmektedir. KIO ile beraber birim/kategorisel bazlı enerji tüketimi %50'ye yakın düşerken toplamda %2-%3 seviyesinde enerji tasarrufu sağlanmaktadır.

Yakın gelecekte birçok cihaz birbirine bağlanıp veri alışverişinde bulunacaktır. Bu bağlamda veriler hızlı bir şekilde akacak olup IOT uygulama ve çözümlerinin bu veri akışına hızlı bir şekilde cevap vermesi gerekmektedir. KIO, yeni versiyonu ile birlikte IOT protokollerinden olan MQTT (Message Queuing Telemery Transfer) protokolünü destekleyecektir. MQTT, TCP/IP üzerinden mesaj alıp göndermeyi sağlayan aracı (broker) temelli hafif bir protokoldür [9]. TLS/SSL seçeneği ve kimlik doğrulama sayesinde IOT güvenliğini de sağlamaktadır. KIO, MQTT'nin kullandığı tüm güvenlik seçeneklerini de barındıracaktır.

KIO, yeni versiyonuyla beraber bir diğer IOT standardı OPC UA (Open Platform Communivations Unified Access) standardını destekleyecektir. Endüstriyel otomasyonda kullanılan donanımların kendilerine ait bir sürücüsü bulunmaktadır. Bu donanımlara erişebilmek için her defasında bir sürücü yazılımına ihtiyaç duyulmaktadır. OPC UA sayesinde bu durum ortadan kalkarak tek bir standart halinde veri iletişimi sağlanır ve üretim hatlarında kullanılan her türlü donanıma ait veriler bağlantı uyumsuzluğu olmaksızın sistemlere aktarılmaktadır [10}. KIO, OPC UA eklentisi ile beraber sahada yer alan her türlü donanım, PLC (Programmable Logic Controller) gibi ögelerden veri toplayabilir hale gelecektir.

Sahada yer alan tüm cihazların birbirleriyle haberleşip veri gönderme işlemi büyük bir miktarda ağ trafiği üreterek veri yönetimine neden olacaktır. Sınır bilişim (edge computing) denilen bu yeni oluşum, cihazların ürettikleri veriyi uç merkezde toplayıp optimize eder, ardından bu veri, buluta ya da veri merkezine aktarılarak performans

artışına neden olmaktadır [11]. KIO, yeni versiyonla beraber KIOBox ürünü ile bu uç merkezlerden biri olacak olup sahadan gelen verileri optimize ederek tüm veriyi ana sunucuya göndermek yerine belli süzgeçten geçmiş veriyi ana sunucuya aktaracaktır. Böylece ana sunucu büyük bir ağ trafiği altında kalmayacak olup performanslı bir şekilde çalışmış olacaktır.

KIO, IOT'nin getrdiği yeniliklere adapte olmakla birlikte, ısı haritaları, sankey diyagramlar, regresyon analizleri gibi yeni nesil IOT uygulamarının özelliklerine de sahip olacaktır.

Kaynakça

1. Doğan H., Yılankıran N.; Türkiye'nin Enerji Verimliği Potansiyeli ve Projeksiyonu, Gazi Üniversitesi Fen Bilimleri Dergisi Part: C, Tasarım ve Teknoloji, s. 375-383 (2015).
2. ISO50001 Enerji Yönetim Sistemi, http:// www.emo.org.tr/ekler/1e9a35c8a1d9357_ek.pdf, son erişim 2019/06/12.
3. Üstünsoy F., Sayan H. Hüseyin,; PLC destekli SCADA ile enerji yönetimi için örnek laboratuvar çalışması, Politeknik Dergisi, 21(4):, s. 1007-1014 (2018).
4. Schneider Electric EcoStruxure, https://www.se.com./tr/tr/work//campaign//innovation/overview.jsp, son erişim 2019/06/14.
5. Siemens MindSphere, https://siemens.mindsphere.io/en, son erişim 2019/06/15.
6. PTC ThingWorx, https://www.ptc.com/en/resources/iiot/product-brief/thingworx-platform, son erişim 2019/06/16.
7. KIO, https://www.kioplatform.com/tr/projeler/bankalar, son erişim 2019/06/17.
8. Akbaş B., Kaya D., Eyidoğan M.; Bir Otomobil Montaj Fabrikasının Enerji Tüketim Analizi ve Enerji Tasarrufu Potansiyelinin Değerlendirilmesi, Mühendis ve Makina Dergisi, TMMOB Makine Mühendisleri Odası, Cilt 59, Sayı 691, s. 85-100 (2018).
9. Haberleşme Protokollerinde Endüstri 4.0 Devrimi | MQTT, https:// www.endüstri40.com/haberlesme-protokollerinde-endüstri-4-0-devrimi-mqtt/, son erişim 2019/06/22.
10. OPC'de Özgürlük; OPC UA, http://otomasyondergisi.com.tr/arsiv/yazi/opcde-ozgurluk-opc-ua/, son erişim 2019/06/22.
11. Edge Computing Nedir? Cloud Computing ve Fog Computing'den Farkı Nedir?, https://www.karel.com.tr/blog/edge-computing-nedir-cloud-computing-ve-fog-computing-den-farki-nedir, son erişim 2019/06/23.