Computer Standards & Interfaces 62 (2019) 1–16

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Application of a software agility assessment model – AgilityMod in the field

Özden Özcan-Top[a,*], Onur Demirors[b,c]

[a] Dundalk Institute of Technology, RSRC, and Lero, Dundalk, Ireland
[b] Department of Computer Engineering, Izmir Institute of Technology, İzmir, Turkey
[c] School of Computer Science and Engineering, University of New South Wales, Sydney, Australia

ARTICLE INFO

Keywords:
Agile software development
Agility assessment
Reference model
AgilityMod

ABSTRACT

Adoption of agile values and principles and transformation of organizations towards agility are not easy and straightforward. Misinterpretation of agile principles and values, and adoption of partial solutions with few agile practices instead of holistic approaches prevent organizations to obtain full benefits of agile methods. We developed the Software Agility Assessment Reference Model (AgilityMod) for the appraisal of software projects from agility perspective and to provide guidance on specifying gaps on the road towards agility (agile maturity). The meta-model of AgilityMod was defined in relation with the ISO/IEC 15504-Process Assessment Model. AgilityMod was developed in an iterative and incremental manner by running successive case studies and getting opinions of experts for the evaluation and improvement of the Model. The multiple case study that we present here in detail included the implementation of the Model in eight software development companies. The results of this case study were evaluated by the case study participants. According to the significant majority of the case study participants, AgilityMod achieves its purpose.

1. Introduction

The foundations of agile methods and the underlying reasons behind agility comes from the values and principles defined in Agile Manifesto [1]. Widely utilized agile methods have been developed since the publication of the Agile Manifesto in 2001. eXtreme Programming (XP) [2], Scrum [3], Feature Driven Development [4] and Crystal methods [5] can be named among the pioneers of agile methods. However, adopting agile methods is not easy or straightforward [6]. As the values and principles in the manifesto provide a general perspective for agility, the derived assumptions bring limitations in agile software development [7]. The majority of agile methods have a descriptive nature which also leads to misinterpretations that sometimes cause to use agile as an excuse for being undisciplined [8]. Based on Schwaber's report (2007), Laanti et al. [9] states that "*Agile adopters are often not aware of what agile adoption really means and how broad a change is required*". The 1000-participant-survey performed by Laanti et al. [9] highlights mainly the following challenges in the agile adoption: *i)* Achieving the right balance between being adaptive and predictive, *ii)* achieving consistency while being flexible, *iii)* losing the bigger picture while working at the iteration level.

To manage adaptation challenges, software organizations seek assistance in practice [10]. One form of such assistance can be supplied by means of structured assessment models or frameworks. Application of structured assessment models can enable identifying where projects stand in terms of agility, and by doing so, depict which areas need improvement. Such an approach for improvement can provide to the point interpretation of agile principles stated in the Agile Manifesto for the specific organizations, and enable more efficient adaptation of agile practices. However, it should also be noted that the ultimate goal for adopting agile methods for an organization should not be "being agile", but finding ways to improve performance, code quality, customer and employee satisfaction through the implementation of agile methods/practices.

We developed the Software Agility Assessment Reference Model (AgilityMod) [11] to assess agility levels of software projects. The Model provides guidance in identifying the state of projects with respect to agility, and depicts improvement opportunities in relation with the agile practices. The meta-model of AgilityMod was defined utilizing the principles defined in ISO/IEC 15504-Process Assessment Model (PAM) [12].

In this paper, we present the results of the application of AgilityMod in eight software development organizations. We aimed to answer two research questions: **RQ1:** How suitable is the Software Assessment Agility Reference Model to be used for identifying software aspects' agility, identifying the agility gaps, and providing roadmaps for

* Corresponding author.
E-mail addresses: ozden.ozcantop@dkit.ie (Ö. Özcan-Top), onurdemirors@iyte.edu.tr (O. Demirors).

https://doi.org/10.1016/j.csi.2018.07.002
Received 5 October 2017; Received in revised form 12 July 2018; Accepted 13 July 2018
Available online 18 July 2018
0920-5489/

improving in agility in a software project? **RQ2:** What are the strengths and weaknesses of AgilityMod?

The cases that we applied the Model included projects from different domains such as technical media, durable consumer goods, ERP solutions, multimedia solutions and e-governance solutions. The assessed projects' team sizes ranged between six employees to 45 employees. The shortest iteration length for the cases was seven days while the longest one was 45 days. In four of the eight cases, distributed software development environments were established. In three of the cases, the development was maintained with internal customers.

After the conduction of each case study, we developed detailed assessment reports indicating the agility gaps, successful practices observed in the cases, and the improvement suggestions. We shared these reports with the organizations and presented the results to the assessment teams and the managers of the organizations. After and/or during each presentation, we discussed the results with the attendees. Following each presentation, we asked participants to fill in the questionnaire we provided which were designed to understand the capability/success of the Model to achieve its purpose. Overall, we obtained feedback from 20 people.

The rest of the paper is structured as follows: In Section 2, we provide the background that leads us to work on this topic and the studies that we performed to evaluate existing agile maturity/assessment models, tools and surveys. In Section 3, we provide the research method used in the study. In Section 4, we describe AgilityMod. In Section 5, we present the multiple case study in detail with the solutions we provided to mitigate the validity threats. We also discuss the capability and the validity of AgilityMod based on the defined criteria. In Section 6, we present the conclusions and future work.

2. Related research

The section starts with background information explaining the agility definition and agile adoption challenges. In Section 2.2 and Section 2.3, we provide a review on agility assessment/agile maturity models, tools and surveys. We also discuss the applicability of process maturity models SPICE [13] and CMMI [14] within the context of agile organizations. However, we left out the agile integrated (hybrid) models in the literature review. The purpose of the hybrid models (such as agile integrated MDevSPICE [15]) is to fulfill specific requirements using agile software development practices. These models do not intend to assess the agility of software projects specifically, but, assess the conformance to regulatory requirements and /or improve software quality/safety with agile integration.

2.1. Background

AgilityMod aims to assess the "agility" of software projects. Boehm and Turner [16] define agility as being innovative, and flexible and adaptive to new environments. Conboy and Fitzgerald's [17] comprehensive review on many disciplines for agility revealed that the agility concept was first introduced in the manufacturing domain in 1991 by a group of researchers at the Iacocca Institute in Lehigh University. In this study (2004), Conboy and Fitzgerald define the agility as "the continual readiness of an entity to rapidly or inherently, proactively or reactively, embrace change, through high quality, simplistic, economical components and relationships with its environment". Erickson et al. [18] describe the agility as follows: "agility means to strip away as much of the heaviness, commonly associated with the traditional software-development methodologies, as possible to promote quick response to changing environments, changes in user requirements, accelerated project deadlines and the like". In 2009, Conboy develops a definition of agility by reviewing the literature across a number of disciplines. His definition of agility is: "the continual readiness of an information systems development method to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment". We define agility in our model as the capability of the practices of a software project in achieving the agile principles defined in the Agile Manifesto.

Agile software development methods are adopted by software communities as they are perceived as solutions for the problems such as missing deadlines, exceeding budgets, delivering final products that do not meet the needs of customer [19]. Laanti et al. [9] performed a survey study to understand the perception of the impacts of agile transformation when it is deployed in a very large software development environment. The survey was performed in Nokia company with more than 1000 participants in 2010. 24% of the participants thought that process tailoring and process improvement were the areas that they experienced most challenges. Requirements management, iterative planning and effort management were other top three perceived challenges. The results show that despite observing challenges, 60% of the survey responders wanted to stay in an agile mode of working, while only 9% wanted to go back to traditional working methods. However, 31% of respondents did not see any difference or did not have a clear opinion. It was also stated that long experience with non-agile development negatively affects certain opinions of agile development. This is a significant finding that might be the root cause of the issues related to cultural changes required in agile adoption.

Based on the 12th VersionOne [20] state-of-agile survey results published in 2018, 84% of respondents stated that their organization was at or below a "still maturing" agility level. Compared to the 11th state-of-agile survey [21], a decrease was seen in respondents citing "organizational culture at odds with agile values" and "lack of business/customer/product owner availability" as challenges for adopting and scaling agile. However, increases were observed on the "fragmented tooling", "inconsistent processes across teams" and "general resistance to change" challenges. Not all survey studies which were performed by commercial companies, follow a scientific rigor. We also accept that there might be bias on the surveys' results, as they were filled out mostly by agile practitioners. However, we believe that large-participant surveys can still be used along with other scientific studies, as they point out the challenges regarding the agile software development and adoption from practitioners' point of view.

Ambler explored the five most common software development methods: Lean, Agile, Iterative, Ad-Hoc, and Traditional, in the 2013 IT Project Success Rates Survey [22]. The survey was performed with 173 participants. The *Lean* approach was found as the most successful one among other strategies for the delivery of products and meeting projects' success criteria. The *Iterative* and *Agile* approaches followed this with a 5% percentage decrease. What noticeable in these results was that Agile and Iterative approaches have approximately the identical results with 64–65% success, 28–30% challenge and 6–7% failure rates.

As mentioned in the 10th State of Agile Report performed by Ambler [23], *organizational culture* continues to dominate the top causes of failed agile projects. Lack of management support for cultural agile transition was reported by 38% of the participants. It was stated that the concerns on required cultural change increased from 44% in 2014 to 55% in 2015. The increase might also be explained with the increase of agile adoption in organizations. *Pre-existing rigid/waterfall framework* and *not having enough personnel with agile experience* were two salient ones among the barriers for further agile adoption. These two barriers also show the importance of guideline use with alternative agile adaptation options.

Ambler stated in his book [19] that there are increasing numbers of project failures associated with agile strategies. He also mentioned that agile and iterative projects produced similar statistical results in terms of quality, success in deliveries, and return on investment in 2008 and 2011 IT Project Success surveys [19]. Both failure stories and identical

Table 1
The criteria used in evaluation of existing models.

Decision points from Maier et al. [25]	Evaluation criteria and their description
Audience: Define expected users of the model **Aim:** Define the aim of the model	**Fitness for Purpose***:* An agile maturity/assessment model should be developed with the purpose of assessing agile process capability and assisting organizations in software process improvement.
Scope: Clarify the scope. Is it generic or domain-specific? **Select Process Areas (Content):** An effective assessment should be based on an underpinning conceptual framework, generated from (traceable) principles of good practice. Selection process (areas) that yield conceptualizations of the field.	**Completeness:** An agile maturity / assessment model should address all or a subset of major engineering and management processes within a software development life cycle. It should include process related definitions, goals, practices or process success indicators which enable assessment of the agile processes.
Define Success Criteria: Success criteria need to be determined at the outset and manifest in the form of high-level and specific requirements.	***Objectivity:*** Verifiable results must be produced. The judgment of the assessor should be at a minimum level. ***Correctness:*** All model elements must be compatible with agile principles. Descriptions, goals and work products should correctly present the related process or process area. ***Consistency:*** An agile maturity model/framework should be internally consistent. There shouldn't be logical or temporal conflicts between two specified model elements.
Select Maturity Levels (Rating Scale): Define a set of maturity levels.	***Definition of Agile Levels:*** An agile maturity/assessment model should provide definitions of agile levels which enumerate the different degrees of agility. Those maturity levels could be interpreted intuitively and should be designed to complement each other.
Validation: Test the populated grid for validity and relevance. **Verification:** Evaluate the developed grid against the success criteria and requirements defined.	***Verified and Validated:*** An agile maturity / assessment model should be verified and validated. This criterion is included in the updated version.

success rates of agile and iterative software development projects provide insight that organizations do not get full benefit from agile software development methods.

2.2. Agile maturity/Assessment models

Structured approaches such as maturity/capability/assessment models aim to assist the transition of organizations towards a specific direction by enabling to capture the status and by identifying the gaps between the current state and the desired state.

In [24], we performed a multiple case study in a software organization to evaluate the capabilities of the existing agile maturity/assessment models based on six criteria. The defined criteria were mainly associated with the study of Maier et al. [25] on development of organizational maturity grids. They reviewed a wide range of maturity and capability models and suggested a roadmap to develop maturity grids with specific decision points. For criteria selection, we adopted these decision points in our study along with our experiences on software process improvement [26,27]. Table 1 below lists the referenced decision points (i.e. audience, aim, scope, etc.), the criteria we used, and their descriptions.

In order to identify the models to be included in the evaluation, we performed a search on the IEEE Explorer, Web of Science, and SpringerLink research platforms, and also on Google Scholar. We listed the models found in these research platforms in Table 2. During the search, we observed that the models published with assessment purpose extensively use the “maturity” keyword. Among nine models found, the first five models were decided to be included in the scope of the study with the following inclusion criteria:

(1) Description of the model should have been given to enable detailed analysis.
(2) The study should have been published in conference proceedings or journals, which is an indicator of academic perspective of the model.

Table 2
Agile maturity models found in the research platforms.

ID of the model	Name of the model	Model owner
M1	Agile Maturity Model [28]	Patel and Ramachandran
M2	Scrum Maturity Model [29]	Yin
M3	Agile Adoption Framework [8]	Sidky
M4	Benefield's Model [30]	Benefield
M5	Agile Scaling Model [19]	Ambler
M6	Agile Maturity Model [31]	Humble and Russel
M7	Simple Life Cycle Agile Maturity Model [32]	Malic
M8	Agile Maturity Model [33]	Proulx
M9	Agile Maturity Model [34]	Jayaraj

The first five models were implemented in the multiple case study research format in a small-sized software development company at which agile practices have been applied for about 1.5 years. The details of this evaluation were provided in [24]. Briefly, among all evaluated models/frameworks, Agile Adoption Framework (**M3**) obtained the best assessment results. However, none of these models satisfied all of the expected criteria. They are needed to be improved in terms of completeness, definitions of agility levels, and objectivity criteria. One of the most prominent difficulty of the models was that their coverage of the software development life cycle. They usually cover part of the life cycle. For instance, **M1** lacks agile practices to cover configuration management and change management processes. Similarly, there are no practices for testing and configuration management in **M2**. The solutions provided by **M3** does not cover all of the agile principles. The second critical problem is related with the depth of the description. A good assessment model should achieve a good balance while defining the underlying reference model. At one hand it has to be detailed enough to enable reliable, reproducible assessments with its components, and on the other hand, it has to be abstract enough to enable different agile methods to be covered. The defined attributes in an assessment model provide guidance to the practitioners and are essential for improvement. **M1, M3** and **M4** are not detailed enough to provide reliable and reproducible assessments. In **M1, M2, M4** and **M5**, the maturity/agility level descriptions are not detailed enough to conduct a complete assessment and to use the assessment results for guidance in process improvement.

Several models have been developed after we have performed the case study presented here. Scaled Agile Framework for Enterprise (SAFe) [35] is a model developed based upon agile and lean principles to be utilized for adopting agile and lean practices in large enterprises. In 2015, a maturity model for SAFe (SAFe MM) has been developed [36]. The SAFe MM was built upon the AAF (M3) of Sidky. The purpose of the SAFe MM is to assist organizations in adopting the SAFe model through stages of maturation paths. Thus, it could achieve the “fitness

for purpose" criterion for adopting the SAFe model. The defined maturity levels of the Model depend on predictable models of agile adoption in an organization. These levels and the suggested practices for each level have been evaluated by the developers of the model in a large-scale software development organization. The assessment results provided guidance on which areas to focus in terms of adopting the SAFe. An objective evaluation of agility levels and the fitness of the practices suggested for each level require further case studies to be performed. However, this model is significant to introduce agile maturity initiatives for large scale software development organizations.

The Agile Compass model which was developed by Fontana et al. [37] in 2015, aims to identify the position of the software development teams on the road to agile maturity, based on seven outcome categories. These categories are: *Practices learning, team conduct, pace of deliveries, features disclosure, software product, customer relationship and organizational support*. This approach measures the progress of the team in each of these categories, which can be thought as levels. It was evaluated and validated based on the interviews performed with nine agile software development teams by Fontana et al. The progress in each category was determined using a checklist. Based on a high-level analysis, Agile Compass achieves the *fitness for purpose, agility levels, verification and validation criteria*. However, we couldn't access neither the links of the checklist and model provided in [38] as they were broken. Therefore, we weren't able to find enough evidence to evaluate the correctness, completeness, objectivity and consistency criteria.

2.3. Agility assessment tools and surveys

In [39], we evaluated agility assessment tools with a multiple case study. As the purpose of the study was to evaluate if current agility assessment tools were sufficient to meet the expected criteria, we excluded the text-based checklists, questionnaires, and frameworks. We found 37 assessment tools that partially automate the assessment process after the literature review. Among them, 11 tools that were listed in Table 3 were open source. The criteria we used for the evaluation included (a) coverage, (b) availability, (c) guidance capability, (d) assessment recording, (e) automated reporting, (f) comparability, (g) different modes of usage, (h) different scopes and (i) extensibility.

The tool that is able to meet the most of the criteria, was the Comparative Agility, with completely satisfying seven (a, b, c, d, e, f, and i) out of nine criteria. However, some tools proved themselves useful for special contexts. For example, Depth of Kanban is useful for assessing Kanban implementations, Enterprise Agility Maturity Matrix is useful for agile transformations. Agile Health Dashboard is useful for monitoring capability of agile teams on sprint basis, and IBM DevOps Practices Self-Assessment is useful to adopt a predefined agile adoption path. Among those tools GSPA: A Generic Software Process Assessment Tool (**T10**) was developed using AgilityMod, CMMI and ISO 15504 models.

Table 3
Agility assessment tools included in the case study.

ID of the tool	Name of the tool	Owner
T1	Agile Enterprise Survey	Storm Consulting
T2	Agile Health Dashboard	Len Legastee
T3	Agile Journey Index	Bill Kerbs
T4	Agile Process Assessment Tool	Info Tech Research Group
T5	Agile Self-Assessment	Cape Project Management
T6	Agility Questionnaire	Marcel Britsch
T7	Comparative Agility	Mike Cohn and Kenny Rubin
T8	Depth of Kanban	Christophe Achouiantz
T9	Enterprise Agility Maturity Matrix	Eliassen Group
T10	GSPA: A Generic Software Process Assessment Tool	Ozan Raşit Yürüm
T11	IBM DevOps Practices Self-Assessment	IBM

The results of this case study showed that none of the current agility assessment tools was able to meet all of the criteria. The majority of the tools use a set of agile practices to indicate the level of agility. While these practices are crucial for specific implementations of agile methods, the mere absence or presence of these practices is not sufficient to indicate the success of the adopted agile method. In addition to that, majority of the tools do not provide an indication of agility levels or the possible improvement areas towards agility.

In addition to the models and tools, we evaluated eight widely used and most frequently referenced agile maturity assessment surveys by means of a case study [40]. The criteria used, which were determined with an exploratory case study, were Comprehensiveness, Fitness for Purpose, Discriminativeness, Objectivity, Conciseness, Generalizability, and Suitability for Multiple Assessment. The results showed that none of the agile maturity self-assessment surveys has fully satisfied the defined criteria. Especially, there was no survey meeting the expectations fully in terms of comprehensiveness, fitness for purpose and suitability for multiple assessment.

Schweigert et al. performed a study to compile current available maturity models [41,42]. They evaluated if a commonly accepted agile maturity model exists, and how the mapping of such a model to Capability Maturity Model Integration (CMMI), ISO/IEC 15504 Part 2 and Part 5 would be. Among 40 maturity models, they included 30 models to the scope of the study. They grouped the models into three: those were using a leveling structure similar to CMMI leveling, those were not having a leveling structure at all and those did not use explicit levels. In the end, they emphasized the gap of a scientific research on this topic and the fact that none of the models fulfills the requirements of ISO/IEC 15504 Part2, performing an assessment.

In 2012, Schweigert et al. conducted a survey to identify what an Agile Maturity Model (AMM) would deliver to its users with 67 participants [43]. According to the results of the survey, most of the participants think that an AMM should measure the perfect implementation of agile practices and organizational support for implementation of them. Another significant result was that more than 65% of participants think that an AMM should distinguish technical, project and organizational level processes and allow individual improvement of each process rather than a simple binary result that the organization is agile or not.

Several other studies in the literature evaluated the applicability of process maturity models, SPICE and CMMI within the context of agile organizations [44–50]. The results showed that although in principle these models are not totally incompatible there are inherent difficulties that cause major difficulties in practical applications. The systematic review performed by Silva et al. [51] lists 81 studies, which use agile methods with CMMI. In this review, they conclude that agile methods have been used by companies to reduce the efforts to reach CMMI Level 2, 3 and 5. Although the companies reached the targeted CMMI levels with agile implementation, agile methods alone were not sufficient to obtain the desired maturity and additional practices were necessary to implement. In another systematic review study, Torrecilla-Salinas et al. [52] evaluated the feasibility of using agile methods to achieve a certain maturity level of the CMMI-DEV for an organization developing Web systems. As stated by McMahon [53] the purpose of agile integration in CMMI is to provide alternatives to traditional approaches to implement CMMI practices that also help process improvement. In 2013, we also evaluated applicability of CMMI and agile software development methods with the evidences from the literature. The purpose of this study was to discuss the myths, which limit the use of these two

approaches together by providing counter examples from the literature [54]. We identified artifacts from agile environments that could be could be used as evidences in the CMMI appraisals. We found common grounds, but also inherent application difficulties.

Common results of the studies depict that there are no commonly accepted agile assessment or agile maturity model in the software community. These results directed us to develop a well-structured agility assessment model, which is consistent with the agile values and principles, to provide a complete guidance on agility adoption covering the whole software development life cycle.

3. Research method

Becker et al. [55] evaluated the design approaches of a large number of maturity models developed in Information Technologies (IT) and stated that *"the procedures and methods that led to these models have only been documented very sketchily"*. They developed a generally applicable model to develop maturity models for IT management using Hevner's [56] design science approach. Their model includes the following stages [55]: **A1:** Problem definition and identification of problem relevance; **A2**: Comparison of existing models; **A3:** Determination of development strategy; **A4:** Iterative maturity model development; **A5:** Selection of the different forms that the targeted communication of the maturity model can take (document-based check-lists, manuals or software tool supported models); **A6:** Making the maturity model accessible to defined user groups; and **A7:** Evaluation to understand whether the maturity model provides the projected benefits and an improved solution for the defined problem.

AgilityMod is an innovative problem-solving artefact and was developed in conformance to the stages presented above. At the A1-A2-A4 and A7 stages, we used qualitative research methods. Creswell [57] states that in qualitative research, researchers collect data in the natural settings through the overview of the documents, observing the behaviour or interviewing the participants. Data can be collected from multiple sources and the research process flows from forth to back and back to forth until a comprehensive model is developed [57]. Examples of qualitative research methods include "action research, case study research, ethnography and grounded theory" [58]. Yin [59] describes the case study method as "an empirical inquiry that investigates a contemporary phenomenon within its real-life context especially when the boundaries between phenomenon and context are not clearly evident". Maier [25] and Hevner [56] state that the case study is one of the design evaluation approaches and can be used to validate the models in the field. Rohloff [60] and Becker [55] used case studies to validate their maturity models. Single and multiple *case studies* have been the main component of our research to state the research problem at hand by evaluating existing models, and to evaluate the applicability of the versions of AgilityMod in real-life settings. These case studies showed *exploratory, explanatory and improving* characteristics based on the Robson's research methodology classification schema [61]. It was stated by Becker et al. that the case study is a valid method to illustrate applicability of developed models [55].

Below, we provided a list of the steps in our research associated with the relevant stages discussed above. The followed steps are also given in Fig. 1.

(1) Problem identification (refers to A1 and A2):
   a) A literature review on current agile maturity/assessment models.
   b) Evaluation of the selected agile maturity models' applicability, strengths and weaknesses with a **multiple case study** in a software organization (the 1st Case Study-was described in Section 2.1 and was published in [24]).
(2) Development of the solution (refers to A3 and A4):
   a) Development of the *aspect dimension* of AgilityMod based on the findings of the 1st case study and the literature on agile software development methods.
   b) Development of *agility dimension* of AgilityMod by exploring the stages of organizations in agile adoption. At this stage, we released AgilityMod v1.0.
(3) Refinements of the solution (refers to A4 and A7):
   a) 1st Refinement: Evaluation of the applicability of AgilityMod v1.0 and discovery of the improvement opportunities on the Model through **a single case study** performed in a software development organization (the 2nd Case Study- Upgrade to AgilityMod v2.0) [62].
   b) 2nd Refinement: Review of AgilityMod v2.0 by agile practitioners, getting detailed feedbacks from them and the refinement of the Model. (Upgrade to AgilityMod v3.0).
(4) Validation of the solution (refers to A7):
   Application of the Model in 8 software organizations by means of a **multiple case study**. The case study included formal process assessments based on AgilityMod v3.0 through semi-structured interviews and evaluation of the project artefacts. Presentation of assessment results to the assessment participants and receiving feedback from the participants.
   In this paper, we focus on the results of the cases studies of this stage. Accordingly, Section 5 includes detailed explanations on how the multiple case study was designed, conducted and analyzed.
(5) Improving Reliability (refers to A7):
   Implementation of the Model in three software organizations by the assessment team members who had not participated in the development of the Model.

4. The model: AgilityMod

AgilityMod's meta-model is defined in accordance to the ISO/IEC 15504-Process Assessment Model (PAM) [12,13]. The purpose of using ISO/IEC 15504's meta-model is to create a common basis for performing assessments of agility, providing a better guidance with standard model elements and presenting the assessment results using a common rating scale.

Even though AgilityMod has inherited the ISO/IEC 15504[1] structure, it has its own characteristics for the name of the dimensions as well as the underlying characteristics. AgilityMod defines the Aspect and the Agility dimensions for the Process and Capability dimensions, respectively. A brief description of the dimension elements and the underlying logic are given below. For a detailed information on AgilityMod please see [11].

4.1. Aspect dimension elements

4.1.1. Aspect

Formal process layers of traditional software development are intertwined to each other in agile software development. It is difficult to specify boundaries of agile processes due to continuity. Aspects, which are the modularization of agile processes and practices are integrated under meaningful and agile compatible abstract definitions. They are sets of interrelated and interacting activities. Four aspects were defined in AgilityMod to cover the software development life cycle: *Exploration, Construction, Transition, and Management*. The Aspects, which also refer to the group of phases in software development, were inspired from the Ambler and Line's Disciplined Agile Delivery (DAD) method [19]. The activities in DAD are handled in three categories: *Inspection,*

[1] The definition of ISO/IEC 15504's model elements differ from the AgilityMod's Model elements. Based on ISO/IEC 15504-Part1, we provide the definitions of two main elements: **Process**: A process transform inputs into outputs with sets of interrelated or interacting activities. **Practice:** A practice is "an activity that contributes to the purpose or outcomes of a process or enhances the capability of a process."

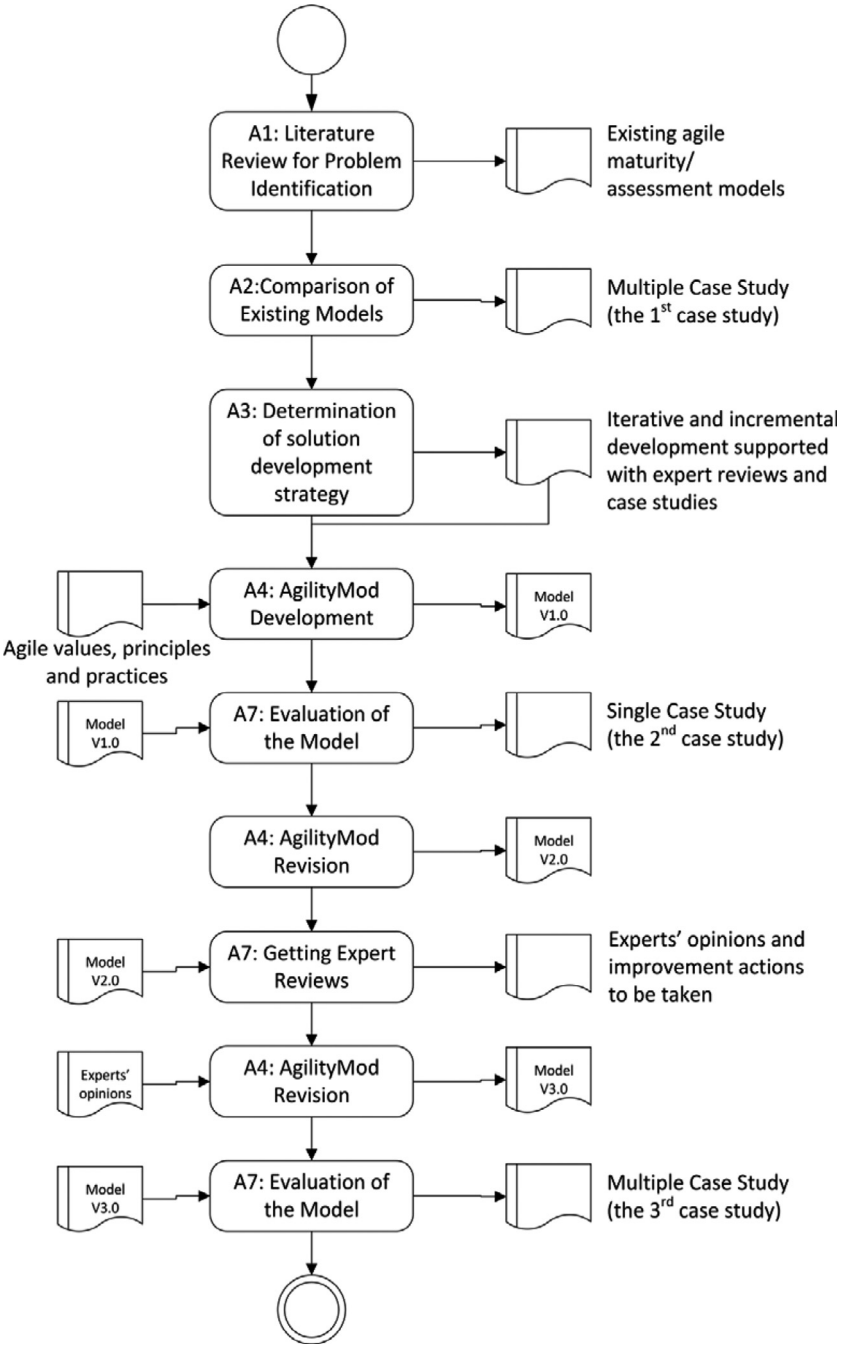

Fig. 1. Followed research steps and their outputs.

Construction and *Transition*. Although all management-type activities are followed within these three categories in DAD, we defined *Management* as a separate aspect as its treatment requires a focus on a different knowledge area. Four aspects in AgilityMod are explained briefly below.

4.1.1.1. Exploration. The purpose of the Exploration Aspect is to understand the customer/user needs, and to transform these needs into artefacts that initiate communication for elaborating them during the construction, and to manage the changes in these artefacts.

4.1.1.2. Construction. The Construction Aspect includes the architecture, design, coding and unit testing activities. The purpose of the Construction Aspect is to develop a high-quality software solution that is ready to be deployed.

4.1.1.3. Transition. The purpose of the Transition Aspect is, to establish and maintain reliable and repeatable build, integration and deployment practices to keep the application in a working state during the development, to obtain feedback about the problems in the process, to make the whole process visible to software development team members and other stakeholders such as customers, and to shorten the response time to changes.

4.1.1.4. Management. The purpose of Management Aspect is to identify, establish and track activities and resources necessary to develop a product. (From agile perspective, the purpose of the Management Aspect is to perform planning and tracking activities continuously and estimating collaboratively to achieve efficiency, and to perform these practices as value adding activities to the project life cycle.) Although the usage of measures by Agile teams are

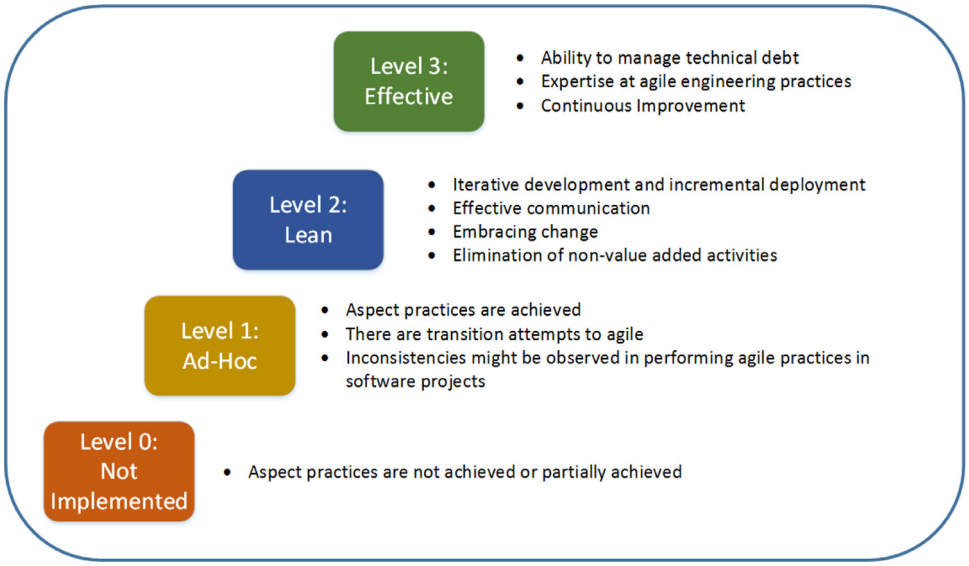

Fig. 2. Agility Levels in AgilityMod.

controversial, the measures (such as Velocity) and estimations (such as Story Points) are parts of Agile methods, and field studies show that agile organizations use measures effectively [63]. Therefore, the management aspect includes measurement related practices as well.

4.1.2. Purpose statement

Purpose statements define the functional objectives of aspects.

4.1.3. Outcome

Outcomes are the expected observable results of the aspect performance. Aspect practices are associated with outcome statements. After defining the purposes of each aspect, we defined the Outcomes of the aspects in line with the purpose statements. The purposes provide general definitions for an aspect; however, the Outcomes provide observable results for the aspects, and guides us to understand what needs to be achieved.

4.1.4. Aspect practice (AP)

Aspect practices are activities or activity groups that contributes to achievement of an aspect purpose and outcome. An activity is something done or, just state of doing something. Each aspect practice is associated with one or more of the outcomes. AgilityMod is a reference model which can be used for any agile method. Therefore, the aspect practices are defined from a generic perspective. An example of the AP is “E.AP1: Capture the customer and user needs”. The Agile Elaborations (AE) describe how standard software development practices can be applied from an agility perspective. At this stage, we associated the agile practices of Scrum, XP, DAD, Agile Modelling and Kanban with APs to identify AEs.

4.1.5. Example work products

Example work products are outputs that can be produced at the end of the successful achievement of an aspect or agility attribute.

4.1.6. Fallacies

Fallacies describe the wrong implementations of agile practices and misinterpretations that need to be avoided.

4.2. AgilityMod agility dimension elements

4.2.1. Agility level (AL)

Agility Levels describe a rational way of progressing through improvement of the agility of any aspect. Four agility levels are described for an Aspect: Level 0: Not Implemented, Level 1: Ad-Hoc, Level 2: Lean, Level 3: Effective. These levels are displayed in Fig. 2. Agility Levels (AL) were initially defined using a common-sense approach based on our earlier research studies [27,64,65] and our experience in implementing the software process improvement models together with Bilgi Grubu Ltd. Sti.[2] [26]. Then, we iteratively updated the model based on the feedback from the case studies and using the opinions of the experts. We reflected the findings to adjust the levels and associated practices for each agility level.

At Level 0, the aspect practices are either “not achieved” or “partially achieved”. At Level 1 (Ad-Hoc), the fundamental development activities such as requirements development, design, coding, integration, testing, and deployment are performed consistently. At Level 1, there are transition attempts towards agility in software projects by exploring the agile practices with trial and discard. Although, the Aspect practices are implemented and Aspect purposes are achieved at Level 1 (this is the main difference from Level:0); there would be inconsistencies in implementing agile practices in projects, which prevent the teams to obtain the benefits of agile approaches. For instance, iterative and incremental development is one of the main agile practices. But, we don't expect software development teams to maintain consistency in performing such practices at Level 1. The Ad-Hoc level should be a temporary stage in agile adoption which is characterized as an undisciplined software development environment partially using agile practices. The counter evidences listed in [11] are the main signs of Ad-Hoc implementation. The name of Level 2, “Lean” was inspired from the following agile principle: "*Simplicity–the art of maximizing the amount of work not done–is essential* [1]." The Aspects at Level 2 are characterized with two attributes: “Iterative” and “Simple”. An Aspect at the Lean level means that utilizing agile practices, it is optimized to deliver working software. At Level 2, work products are developed iteratively so that frequent feedback are provided to and obtained from the customer to improve the product capabilities, to identify bottlenecks and problematic areas, and to surface assumptions related to the product. The “Simple” attribute eliminates non-value-added processes and technical activities, and also achieves a balance between adaptive and predictive works. Technical improvements are present at Level 2 and above. At Level 3, each aspect is performed to achieve delivering value with high productivity and low defects by employing agile engineering practices and using agile tools to support a continuously

[2] www.bg.com.tr.

Table 4
Aspect attributes of each agility level and generic agility practices.

Agility level	Aspect attribute	Generic agility practice
L1: Ad-Hoc	1.1 Performing Aspect Practices	1.1.1 Perform aspect practices
L2: Lean	2.1 Iterative	2.1.1 Develop work products in an iterative and incremental way
		2.1.2 Communicate effectively
	2.2 Simple	2.2.1 Balance the predictive work and adaptive work
		2.2.2 Employ minimally sufficient ceremony
L3: Effective	3.1 Technically Excellent	3.1.1 Incorporate agile engineering methods/practices to the aspect practices
		3.1.2 Integrate tools to aspects to improve the productivity
	3.2 Learning	3.2.1 Support collaborative work and shared responsibility
		3.2.2 Adopt agile leadership styles and adjust the behaviors towards mistakes of people
		3.2.3 Encourage people in the organization to participate in learning, teaching and improvement
		3.2.4 Collect measures to support learning and improvement

improving environment. Although the experimentation on agile practices starts for "learning and improvement" at Level 1, it is not structured and continuous. At Level 2, we eliminate any kind of activity that does not add value, but cause waste in software development process. At Level 3, the software development teams achieve continuous improvement by themselves without guidance of external coaches.

4.2.2. Aspect attribute (AA)

Aspect attribute is an indicator of the aspect performance. It defines the characteristic of the aspect. They are applicable to all aspect practices. All of the aspect attributes were directly derived from the agile manifesto and its 12 agile principles, by combining the related principles together. At the 1st Level (Ad-Hoc), the aspect attribute is *"Perform aspect practices"*. At the 2nd Level (Lean), the aspect attributes are "*Iterative and Simple*". At the 3rd Level (Effective), the aspect practices are *"Technically Excellent and Learning"*. The mapping between the aspect attributes and the agile principles are given in [11]. The full model is given in [11,66].

4.2.3. Generic agility practice (GAP)

Generic agility practices are the activities or activity groups that contributes to and support achievement of aspect attributes. We provide the aspect attributes and generic agility practices related to each agility level in Table 4.

4.2.4. Generic work product (GWP)

Generic work products are expected evidences as a result of achievement of an aspect attribute. GWPs are limited with the given list in the Model and can be extended based on working environments.

4.2.5. Generic people & resources (GPR)

Generic resource is a kind of resource (equipment, software, hardware, facility, funding, etc.) that is utilized in the conduct of an aspect or agility attribute. As the primary driver of agile software development are humans rather than tools and processes, we separated "people" from the other resources which are also essential but, second order influencers behind success.

5. Application of the model

We performed a multiple case study to investigate if the proposed model can be utilized for agility assessment of software projects, and if it can provide roadmaps to improve their agility. We defined the following Research Questions (RQ) in accordance with the objectives:

RQ1: How suitable is AgilityMod for identifying software aspects' agility, identifying agility gaps, and providing roadmaps for improving agility in a software project?
RQ2: What are the strengths and weaknesses of AgilityMod?

5.1. Design of the case study

The case study was planned to include the following activities:

(a) *Preparation:* Development of the assessment questions, the data collection templates to be used in the assessment, and the survey questions that will be used after findings' presentation.
(b) *Case Selection and Planning*: Interacting with candidate case organizations, deciding on assessment dates and participants with the agreed organizations.
We planned to perform case studies in eight different organizations to increase the generalizability (external validity) of the study. For the selection of the organizations, we aimed to observe all patterns at the aspect and agility dimensions of AgilityMod. Therefore, we selected cases which are at different stages of agile adoption.
(c) *Assessments and Data Collection*: Performing the assessments and data collection via semi-structured interviews and observing direct evidences per project.
We planned to conduct formal assessments (gap analysis) through semi-structured interviews with process owners and evaluate direct evidences per project. People from different roles were planned to be involved in the interviews to obtain tacit knowledge directly from practitioners. These roles were planned to include at least one product owner, one business analyst, one developer, one configuration manager and one tester. Besides interviews, direct evidences such as product/sprint backlogs, daily stand-up meetings, tools used, metrics collected were also planned to be analyzed.
(d) *Analysis:* Developing findings reports by analyzing the notes taken during interviews and voice records for each case.
(e) *Validation of the Findings*: Discussing the findings with interviewees to obtain their feedback on the assessment results and revising the reports accordingly.

5.2. Case study conduct

We applied the model in eight projects from eight organizations. The information about the projects and the organizations are summarized in Table 5. The domains that we applied the Model are technical media, durable consumer goods, ERP solutions, multimedia solutions and e-governance solutions. The team sizes of the assessed projects ranged between six employees to 45 employees. In projects #1, #4, and #8, there was no external customer, the teams specified and analyzed the product requirements by themselves. In projects, #2, #5, #6, and #7, the external customers, who were working in different locations, provided the specifications to the teams via the product owners. In project #3, both internal and external customers specified the requirements. For customers located in different offices, the project teams established various solutions to improve communication such as frequent teleconferencing, customer-side face to face meetings, and maintaining communication matrices for monitoring the efficiency of

Table 5
Demographics of the cases.

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8
Organization's Business Domain	Tech-media company	Government IT organization	ERP solutions company	Various communication systems	Software and Internet Security	Cloud and multimedia solutions	Solution provider in Telco and Finance sector	Durable consumer goods industry
Project/ Team Size	22 full-time	23 full-time	19 full-time	7 full-time, 3 part-time	24 full time	45 full time	6 full time	3 part time, 1 full time
Project Domain	Multimedia Services System	Management Information System	Enterprise Decision Support System	Unknown	Digital Advertisement Operations Platform	Voice and Visual Communication System	Online Self-Care Portal	Internal Project Management Portal
Team Location	Local	Local	Local	Local	Local	Local	Local	Local
Customer Location	Internal development, no direct customer	External, in another location	Both Internal and External	Internal development, no direct customer, COTS product	External, in another country	External, in another location	External, in another location	Both Internal and External
Programming Language	PHP	J2EE, Flex	Pascal	PHP	PHP, Java Phyton, Cassandra	Java, C++, C, Java Script, HTML5	Java	Readmire
Customer Communication Means	Face-to-face	e-mail, phone, face to face	Email, phone over partners, over support portal, face to face	Face-to-face	Tele-conference	Tele-Conference, e-mail	Face-to-face, phone, e-mail	Face to face, Tele-Conference, e-mail, phone
Iteration Cycle Time and Consistency	7 to 10 days Consistent	30 days Consistent	45 days Consistent (7 weeks)	15 days to 30 days No consistency	3 weeks, Consistent	3 weeks, No consistency	3 weeks, No consistency	4 weeks, Consistent
Approximate Code Coverage Percentage	None	Varies based on modules, 54.5% on average	Code coverage over automated tests 23%	None	Approximately %10	None	None	None
Usage of Continuous Integration	None	Applied	None	None	Initiated	Initiated	Initiated	None
Type of Agreement with Customer	None	Fixed Schedule and Price Contract	None	None	None	Fixed Schedule and Price Contract	Fixed Schedule and Price Contract	None

the communication. In the 10th row of Table 5, where we provided the information on usage of continuous integration, we used the "Applied" label to indicate that a CI system was established for the project, whereas we used the label "Initiated" for the cases that the main steps such as defining the strategy, having trainings on CI, and evaluating alternative CI tools were initiated in the project.

After the conduct of each case study, we developed detailed assessment reports indicating the agility gaps and successful practices observed in the cases. We shared these reports with the related case organizations. In addition, we presented the results to the assessment teams, as well as the managers and CEOs of some of the organizations. The presentations covered the assessment findings, agility levels of aspects, and improvement suggestions. After or during each presentation, we discussed the results with the attendees.

Below we present each case briefly.

Case #1: Organization 1 is one of the leading media companies in Turkey with its millions of visitors in various internet platforms. We selected Organization 1, as they are in an agile adoption process for 2 years and we expected to observe the applicability of AgilityMod in a company that recently applied concepts of agility. The organization mainly performs maintenance activities for its released products. We assessed one of the ongoing projects, Project #1, which includes both new development and maintenance requirements. The project (and the product) #1 is an online video platform which has millions of unique visitors and video views in a month. The project team includes nine software developers, two graphical user interface designers, one tester, two business intelligence analysts and eight content providers. In the scope of this case study, we assessed aspects of Project #1 through interviews and evaluation of the project artefacts. The assessment was performed by one of the authors. The overall assessment including observing the project artifacts took three hours performed with two project managers, and the software team leader. For this case, the evaluation included a sample of user stories written for the project, the KanBan board, a sample of the found defects, and the collected metrics.

Case #2: Organization 2 is a government IT organization that develops e-government software for various government agencies. Project #2 is an e-government project providing solutions to a ministry department which has 40 non-profit organizations located in different cities of Turkey and serving to millions of Turkish citizens. The team includes 21 employees divided into four teams which report to a project manager and an assistant project manager. Three of these teams works on development of software modules, and one team deals with system infrastructure. The overall assessment was performed in three hours. The assessment included an interview with the technical leader of the infrastructure team, who had formerly worked as a developer and has knowledge about project's processes. Following the interview, we observed the tools set used for issue tracking, document management, code management, version control, code review and continuous integration and evaluated a sample of epics and user stories, baselined documents, and defect and risk records.

Case #3: Organization 3 serves over one million end users and thousands of companies with ERP solutions. In the scope of this assessment, we evaluated the aspects of an ERP product developed on Windows infrastructure. The project team, which was divided into two teams, includes one product owner, one scrum master, one business analyst, five testers and eleven developers. The assessment was performed with the scrum master, the product development manager and the product owner over skype. The overall assessment and observation of the project artifacts took four hours. We went through the product and sprint backlogs, analyzed the way that the user stories were described and their approach for issue tracking. The assessment findings were presented to the product development manager, test team manager, software development director of the

	1. AD-HOC								2. LEAN				3. EFFECTIVE					
									Iterative		Simple		Technically Excellent		Learning			
Aspects/Practices	**AP1**	**AP2**	**AP3**	**AP4**	**AP5**	**AP6**	**AP7**	**AP8**	**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**
Exploration	2	2	3	2	3	2	-	-	3	3	2	1	1	1	3	3	1	1
Construction	2	2	2	2	-	-	-	-	3	3	2	1	1	2	3	3	1	1
Transition	2	3	3	1	2	2	-	-	3	3	1	2	1	2	3	3	1	1
Management	2	3	1	3	3	2	2	1	3	3	2	2	1	2	3	3	1	1

Fig. 3. Rating of Each Practice of Case 1.

organization, product owner, scrum master, and a tester in a face-to-face meeting.

Case #4: Organization 4 is a solution provider for information and communication technologies in local and global markets. Due to security requirements, the scope of the assessed project was kept confidential. The project #4 includes four software developers, one test engineer, one product manager and three part-time business analysts. There is no specific project manager in the project. Because of high confidentiality in the project, we were not able to observe the project artefacts, but performed a face-to-face interview in a three-hour time with the product manager, the software team leader, and the test engineer, one by one.

Case #5: Organization 5 works on internet security field. They develop products with the purpose of securing information on internet, securing websites, e-commerce applications and personal computers. It's is an international company, doing business in several countries, with its millions of end users, and thousands of business partners. The product #5 is a digital advertisement sharing platform. It is in use, and new versions of the product are being deployed continuously. The purpose of the project is to ensure the security of the advertisements, and deliver harmless and focused advertisements to end users. The project includes 24 employees. There are four testers and thirteen developers, which are grouped into three teams. One program manager, one architect, and three scrum masters serve for the whole team. Apart from these members, there are two product managers, which can be thought as product owners, living in USA and Turkey. We assessed aspects of the project through interviews and direct observation of the evidences. The assessment was performed in 3-hour time with the configuration manager and the quality assurance manager, who is also the scrum master. The interview was performed simultaneous to the project artefact' demonstration, which included going through the product and sprint backlogs, a sample of stories, a tool-based agile board, and the defect tracking approach.

Case #6: Organization 6 develops products in the field of information and communications technologies. The project #6 that we assessed, enables voice and visual communication via web browsers. The project team includes 45 people divided into six scrum teams. The product owner and the customer are based in USA. The product owner joins meetings via teleconferencing. The team includes developers, testers, scrum masters, and architects. The architects also work as business analysts. In USA, there are two solution architects, who are in communication with the product owner. On top all scrum teams, there is a technical project manager. The overall assessment including going through the project artifacts was performed in 4-hour time with the quality manager, the test manager, the solution architect, and the technical project manager. After the interviews, we went through the tools used for requirements and issue management, product and sprint backlogs, and a software design document.

Case #7: Organization 7 is a solution provider in telecommunication and finance sectors. It is a small sized, innovative company, in which agile methods have been in use since 2010. The product #7 is used to manage the services between a service provider and the licensed operators, based on the regulations mandated by Information and Communications Technologies Authority in Turkey. It is an ongoing project. New features are being developed and the maintenance of the old features is performed at the same time. The project team includes one product owner, one scrum master, three software developers, and one quality assurance engineer. We performed the assessment with the scrum master, the test and configuration engineer and the senior software developer in 4-hour time. We were not able to go through any project artifacts in this case.

Case #8: Organization 8 is one of the biggest organizations in durable consumer goods industry in Turkey. We assessed project #8 which is a project management portal to be used by all divisions of the company. The assessment took four hours in total and the interviews were performed with the whole team, which included the product owner, the project manager, the software developer, and the configuration manager. We went through the product and sprint backlogs, a sample of stories, bugs, tasks, software requirements, and design documents.

5.3. Results

We presented the case study results in two sub-sections: First, we analyzed how successful is the Model in terms of achieving its purpose (**RQ1**). Second, we presented the issues related to structure of the Model, and how the model can be improved (**RQ2**).

5.3.1. Analysis of the model's applicability (RQ1)

The teams of the assessed projects deliver working software in different iteration durations. In the agile manifesto, one of the principles is "Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale" [1]. The importance of fast feedback is emphasized in [67] as follows: "*This fast feedback enables teams to quickly prune unfavorable product paths or development approaches before they compound a bad decision with many follow-on decisions that are coupled to the bad decision*". The shortest iteration length was in the Project #1 with 7 days and the longest iteration length belonged to the Project #3 with 45 days. The length of the iteration is an indicator how fast the feedback has been obtained.

In all eight case studies, we evaluated the projects' agility for all aspects of the Model: Exploration, Construction, Transition and Management. Fig. 3 and Fig. 4 below give the colored schemas of the assessment ratings for Case #1 and Case #2 as samples. These figures enable capturing the assessment results at a glance and show where to direct focus in terms of the problematic areas (the detailed assessment results were provided in the technical report [68]). Each column in the Figures refers to the practices of AgilityMod for three levels of agility. The colors and the numbers in each cell refer to the achieved levels of these practices. We used a four-level scale to express the achievement of the aspect attributes: "Not Achieved (0-Red), Partially Achieved (1-Yellow), Largely Achieved (2-Orange) and Fully Achieved (3-Green) and Not Applicable ((-)- White)". For an agility level to be reached, all of the practices should be largely or fully achieved at that level. Based on this rule, the *exploration aspect* of **Case 1** is at Level 1: Ad-Hoc Level. Because it had a *partially achieved* (1-yellow) generic agility practice (*Simple-GP 2.2.2*) at Level 2. As a comparison, the agility level of all aspects of **Case 2** are at Level 3, as all of its practices were either rated as Largely Achieved or Fully Achieved. The achieved agility levels of each aspect for Case 1 and Case 2 can be found on Fig. 5.

One of the rationales for selecting the cases was to observe a rating for each agility level for each aspect, as this would be the indicator of

	1. AD-HOC								2. LEAN				3. EFFECTIVE					
									Iterative		Simple		Technically Excellent		Learning			
Aspects/Practices	AP1	AP2	AP3	AP4	AP5	AP6	AP7	AP8	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
Exploration	3	3	3	3	3	3	-	-	3	2	2	2	3	3	3	3	2	2
Construction	3	3	3	3	-	-	-	-	3	3	3	2	3	3	3	3	2	3
Transition	3	3	3	3	3	3	-	-	3	2	3	3	3	3	3	3	2	3
Management	3	3	2	2	3	3	2	2	3	3	3	2	3	3	3	3	2	3

Fig. 4. Rating of Each Practice of Case 2.

Fig. 5. Achieved Agility Levels of the Aspects for Case 1 and Case 2.

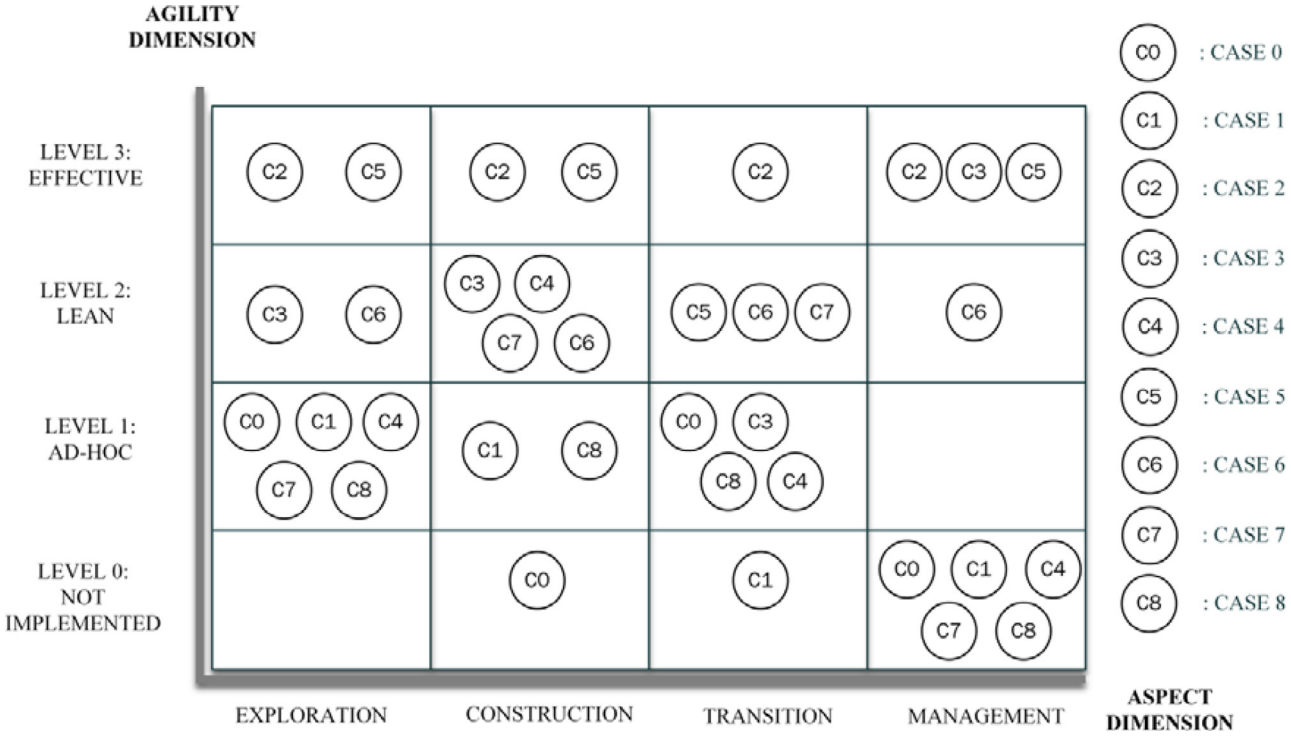

Fig. 6. Distribution of the achieved agility levels.

the logical agility leveling. We presented the distribution of the achieved agility levels for aspects of all eight cases in Fig. 6. We also included the results of the first exploratory case study: Case #0 to the Figure as the assessed project in this case enabled us to observe the lower agility levels of different aspects.

As can be seen in Fig. 6, we were able to observe the occurrence of the agility levels for all of the aspects except for Level 0 of the Exploration aspect and Level 1 of the Management aspect. Observation of most of the agility levels for each aspect shows us the capability of AgilityMod in specifying and representing diversities between the

Table 6
Questions in validation questionnaire.

No	Questions
1)	What is your role in the organization? Could you please describe your background and experiences on agile software development?
2)	Does the report/presentation cover all the improvement areas that you notice about the organization's agile processes? If not, what are the missing ones?
3)	Which of the findings and improvement suggestions presented in the report/presentation have you noticed before? Which of them were new to you?
4)	Does the agility improvement path that is presented to you in the assessment report sound reasonable? Would you prefer the same improvement path? What would be your priorities?
5)	To what extent the presented findings and improvement opportunities in your projects overlap with the issues you have already known in your project? Please select the scale that applies from partially achieved to fully achieved level.

agility levels. This also has been achieved by the broad perspective of the case projects in terms of agile adoption.

The assessment findings were delivered to the assessment team members in the form of assessment reports. Following the findings presentations and discussions, the capability of AgilityMod in achieving its purpose in terms of identifying agility gaps and depicting new improvement opportunities were evaluated by the case study participants. At the end of each presentation, we asked the participants to fill in the questionnaire given in Table 6. All the questions except for the last one in the questionnaire were designed as open-ended questions. The purpose of the last question was to understand the capability/success of the Model to uncover the issues which were unknown to the teams.

After the findings presentation, we asked the case study participants to provide a rating for the fifth question from four scale levels: "Not Achieved (NA), Partially Achieved (PA), Largely Achieved (LA) and Fully Achieved (FA)". Some of the instances were not rated (NR) by some of the participants, as these aspects were not their primary work area and/or they did not feel comfortable in making an objective judgment. These ratings were important for us to understand to what extent we could meet the expectations of the participants in terms of the presented findings and improvement opportunities for their projects.

For eight cases, we obtained feedback from 20 people overall (3, 1, 2, 3, 3, 4, 2, and 2 participants from Case 1 to Case 8, respectively). They rated the achievement of each aspect from their own perspectives. Summary of the ratings are given below:

- Exploration Aspect: 14/20 fully achieved, 3/20 largely achieved and 3 not rated
- Construction Aspect: 14/20 fully achieved, 3/20 largely achieved and 3 not rated
- Transition Aspect: 14/20 fully achieved, 5/20 largely achieved and 1 not rated
- Management Aspect: 15/20 fully achieved, 1/20 largely achieved and 4 not rated

When we took out the "not rated" instances, we had 69 instances in total. From a general perspective, 57 instances over 69 instances were rated as fully achieved, 12 instances over 69 instances were rated as largely achieved. These results indicate that AgilityMod based agility assessment is successful in finding agility issues and revealing agility improvement opportunities in software development projects. At the same time, the model itself has some improvement opportunities as well.

The participants think that the issues provided in the reports covered both previously discovered and undiscovered improvement items from an agile perspective. Their comments indicate that AgilityMod achieves to reveal the unnoticed problems in the software projects. The project manager of Case 1 said that "*the results are very beneficial for us to discover our potential for improvement. The suggestions that were given in terms of the construction and the transition aspects had the highest influence for us.*" The software team leader of Case 2 rated the Transition Aspect at the Largely Achieved Level. He called attention to the need of obtaining feedback from the customers and end users after the delivery of a software product. He mentioned that even if the software has delivered successfully, there might be issues related to the hardware systems and the network and the end product might be perceived as of poor quality because of such issues. He stated that the assessment needs to be extended to analyse the processes of the maintenance phase to monitor the released software. Similarly, the software verification manager of Case 6 thinks that AgilityMod needs to question the practices after software deployment since these are highly correlated with software quality.

The software product development manager of Case 3 mentioned that the presentation covered all the gaps related to their agile processes. He mentioned that they need to invest on the problems related to developers' testing approach, code integration, the need of developers' and testers' work in a collaborative environment and doing more investment on code refactoring and automated tests.

The participants of Case 4 specified that they did not notice the following issues before our assessment and presentation: Improving agility awareness of the project team through agile trainings, establishing dependencies among requirements, establishing a measurement and monitoring infrastructure, managing risks, prioritization of backlog items and estimation of requirements items.

The participants of Case 5 were working on establishing an agile software development culture for 15 months and they had already analysed what their next improvements should be. They mentioned that the assessment results were fully compatible with their own findings. We suggested improvements on technical debt management and advised taking external agile adoption trainings. They decided to update their plans based on these suggestions.

In order to evaluate the usefulness of the leveled approach for improvement, we asked the assessment participants if they would prefer to apply improvement suggestions for each agility stage in the order we specified. The answers we obtained varied. In Case 1, Case 2, Case 4, Case 5, Case 6 and Case 7, the participants agreed on applying the improvement suggestions in the order we specified. However, in Case 3 and Case 8, most participants mentioned that they would not need an order for implementing the suggestions, since they thought the suggestions were independent from each other. Although, these results show independence of the aspects from each other, the evaluation of the effectiveness of the roadmaps provided require follow-up assessments which were not part of this research.

5.3.2. Findings related to the structure of the model (RQ2)

When we evaluated the Model in terms of its components and components descriptions, the following issues emerged during the conduct of the case study:

One of the aims of developing AgilityMod was to minimize the subjective judgment for agility assessment and to identify false agile adaptations. With this purpose, we defined the "Agile Elaborations" for the aspect practices and described the exemplar outcomes that can be observed after successful achievement of generic agility practices (GAP). However, during the conduct of the 1st case study, we observed that the "communicate effectively" GAP in the "Iterative" attribute and the "align with agile values and principles" GAP in the "Management" aspect, require talking to whole teams of the projects rather than an assessment group. We couldn't have the chance to talk with all of the

project team members for these specific practices, however, we extended our question set to be able to understand different ways of communication in projects, and detect communication problems. For other practices, we resolved this problem by asking the same questions to different roles in the project to observe variances in agile perception.

One of the attributes of the 3rd agility level is the "Learning" attribute. In the scope of the "Learning" attribute, we assess how the aspects serve for the purpose of organizational learning and improvement. We aimed to capture the evidences for collaborative work and shared responsibility in the conduct of aspect practices, agile leadership styles, and encouraging people in the organization to participate in learning, teaching and improvement. What we observe in the multiple case study was that, if collaborative work or self-organization was established in a project, it was not just valid for one specific aspect, but valid for the other aspects as well. Therefore, when an evidence was observed during the assessment of an aspect through an agility practice, there was almost no need to question the same agility practice for the remaining aspects as the answer remains to be pretty much the same. In some cases, there might be significant differences in the answers of the 4th agility practice of the learning attribute which is "collecting measures to support learning and improvement". We mention this situation as an issue to be considered, because the improvement suggestions repeat itself in the assessment reports.

It was observed that the positive and negative evidences for agility attributes at the 2nd level may also be the evidences for some of the agility practices at the 3rd level. For example, to employ minimally sufficient ceremony in any kind of activity, we expect aspects to be evaluated regularly for elimination of the redundancies and non-value-added activities. The activities eliminated might be related to technical or process related issues. On the other hand, the purpose of the learning attribute at the 3rd level is to learn from past experiences, and achieve continuous improvement, which also requires performing regular retrospective analysis and taking action. These actions would also require elimination of technical and process related issues. Therefore, the components at the 2nd and the 3rd level may trigger each other, and there may not be clear cut distinctions between these components.

Even if we updated the Model based on the results of the exploratory case study in terms of resolving the overlaps in definitions, during the conduct of the multiple case study we observed that the "Balance the predictive work and adaptive work" generic agility practice, the "Make the artifacts visible to everyone" aspect practice in Exploration aspect, and the "Make the progress visible" aspect practice in Transition aspect overlap. Accordingly, we updated the Model v3.0 by removing visibility emphasis from the generic agility practices and updated the case reports accordingly.

In the 1st version of AgilityMod, there was a 5th Aspect called the "Culture". We removed this aspect from the aspect dimension after the exploratory case study, as its practices overlapped with the "Learning" attribute practices and the "Management" aspect practices. In order to cover the unique practices of culture aspect, we extended other attributes or aspect practices in the Model. During the conduct of the multiple case study, we evaluated if the coverage of the Culture aspect has been fully achieved with the extensions. We have not observed any gaps in the coverage of removed Culture Aspect, as the cultural elements and practices that need to be considered at the project level are covered under the "Learning" Attribute and the "Management" Aspect. The Generic Practices (GP), "GP 3.2.1 Support collaborative work and shared responsibility" and "GP 3.2.2 Adopt agile leadership styles and adjust the behaviours towards mistakes of people" support cultural change in the model at project level. On the other hand, having agility at the organizational level requires not only scaling the software development practices but also changing or adjusting non-IT departments' work approach. Such a scaling would require significant cultural shifts because of the possible organizational structural changes and political challenges. In that case, the "culture" itself can be a standalone aspect.

5.4. Mitigation of threats to validity

The case study approach uses qualitative data and provides solutions in its own context. As a result, some validity concerns might arise in the case study research. We performed the following actions to prevent any threats that could affect the internal, construct and external validity and reliability.

5.4.1. Internal validity

A threat to internal validity is asking diverse questions to participants related to aspect practices during the interviews and getting the answers from them at different granularity levels. In order to prevent this as much as possible, we developed a detailed question set related to each aspect and generic agility practice at the case study design phase.

5.4.2. Construct validity

We perceived two construct validity threats. In the case study, we evaluated the aspects both with conducting interviews and observing the evidences in the form of documents (available records of execution such as plans, stories, e-mails). It is hard to manage the question and answer process in an interview and take notes simultaneously. In order to overcome this challenge, we recorded the assessment process in 7 of the 8 cases, and decoded the records by listening them. For the one that we couldn't record (Case #3), the assessment was performed by two assessors: one took notes while the other one asked questions and managed the assessment process.

The case study analysis phase includes objective evaluation of the results. Each practice has to be rated in an objective way. In order to prevent subjectivity threat as much as possible, we defined possible answers and types of the evidences for aspect practices. We used the four-point ordinal scale of ISO/IEC 15504 to rate the practices, which has one of the values "Not Achieved", "Partially Achieved", "Largely Achieved" and "Fully Achieved" .

5.4.3. External validity

To improve the generalizability of the results, we selected the cases from different business domains which are listed in Table 5. We have not observed any difference or difficulty for the application of the model in these domains.

To be able to observe different agility levels for each aspect, we selected the cases based on agile adoption duration in the company. The cases cover experience on agile software development in the companies from one year to five years.

As AgilityMod has been designed being independent of any specific agile software development method, we aimed to observe applicability of it at different settings. Scrum was the major agile software development method preferred for the management of the projects we assessed. Other methods or practices such as continuous integration and test-driven development were accompanying Scrum to ensure a full software development life cycle coverage in the companies. This allowed us to observe applicability of AgilityMod on other methods other than Scrum.

5.4.4. Reliability

The Model has been evaluated in eight software development companies by one of the developers of the Model as described above. As relevant to reliability, we were concerned with to what extent the success of the Model is dependent on the specific researchers. The Model has been implemented in three other software development companies (different from the ones presented here) in the guidance of another researcher, who has not been involved in the development of the Model [69]. These assessments were performed by the assessment teams established within the companies using the AssessAgility tool which was developed to improve the efficiency of the assessments [70]. They evaluated the capability of AgilityMod's in identifying agility gaps and revealing new improvement opportunities for each aspect based on

Table 7
Evaluation of the capability and the validity of AgilityMod based on the defined criteria.

Criteria	Analysis the model
Audience of the Model: Define the expected users of the Model.	The audience of the Model is agile software development practitioners and coaches. The model was used for agility assessment by assessment teams established within the companies. The assessment teams, the members of which had no knowledge on the model prior to the assessment, included an assessment team leader, software developers, and business analysts. The assessment team leaders guided the interview process.
Aim of Model: Define the aim of the Model which can be either of improvement or benchmarking.	AgilityMod was developed with the aim of assessing agility (agile maturity) levels of software projects and providing guidance in identifying the improvement opportunities regarding agile software development projects in a structured way. The model was implemented in eight organizations by one of the authors of this paper. Twenty participants rated the success of the Model in finding the agility issues and revealing the agility improvement opportunities in software development projects. Significant majority of the participants found the Model successful in achieving its purpose. The Model was also implemented in three more organizations by external assessors to improve its reliability. The case study participants evaluated the Model's success in identifying agility gaps and revealing new improvement opportunities for each aspect. Among the ratings given by the case study participants, 19 out of 24 instances were fully achieved, two of them were largely achieved and three of them were not rated.
Scope: Define the scope of the model which can be generic or domain specific.	AgilityMod is a generic model that can be used in the assessment of software projects with a variety of agile methods in different application domains. Fontana et al. [38] also pointed out that highly customized agile maturity models for specific contexts limit their effective utilization by software development teams, since agile development do not usually rely on standard processes.
Process Areas: Define process areas so that they are mutually exclusive and collectively exhaustive	AgilityMod defines four aspects, which provide whole software development life cycle coverage along with the aspect practices.
Maturity Levels	The Agility dimension provides a mapping of the agility practices with the agile principles. This ensures the achievement of the agile principles and as such the agile values published in the agile manifesto. The agility levels provide the directions towards agility in a manner that they can be introduced and established in the organization.
Verification and Validation:	Verification of the Model was performed by the field experts. It was validated in the field through multiple case studies including 11 cases (counting in three cases which were performed by another researcher independently – described in Section 5.4.D)

a rating scales: *not achieved (NA), partially achieved (PA), largely achieved (LA) and fully achieved (FA) levels.*

A total of six team members from three companies were involved in the assessments. They found the aspect findings successful in finding agility related issues, and found the provided solutions helpful for improvement. Among these ratings 19 over 24 instances were FA, two of them were LA and three of them were Not Rated. These assessments, enabled us to observe that the Model can be easily applied by others and the success of findings do not dependent of the involvement of the developers of the Model.

5.5. Discussion

The endeavor to develop an agility assessment model commenced with the case study, we identified capabilities of existing agility assessment/agile maturity models [24]. In the light of the findings of this case study, we defined the requirements of AgilityMod and developed the Model in an iterative and incremental way. The stages of the Model were conformant to the requirements defined by Becker et al. [55].

We discuss the capability and the validity of AgilityMod based on the criteria defined for developing maturity models by Maier et al. [25] in Table 7.

6. Conclusions

In this study, we presented the Software Agility Assessment Reference Model, AgilityMod, developed for the assessment of software projects from an agility perspective.

It has been developed based on an accepted assessment structure: ISO/IEC 15504, which enables identification of agility issues in a systematic way. It proposes its own components which are characteristic to the agile software development domain such as agility levels, agility practices, resources and outputs. The agility levels and the practices associated with these levels show major variations in the literature. AgilityMod is developed as a practical approach for agility assessment with the feedback obtained in the field by means of successive iterations for a set of case studies. It is detailed enough to be applied consistently and abstract enough to enable different implementation methodologies to be covered. It takes years such models to get mature. We foresee that the Model will be improved through the practitioners' experiences in the field.

AgilityMod have unique attributes which improves its usability and capability with respect to similar models developed previously. We can assess *agility* of a project in terms of four aspects instead of checking compatibility to some agile practices.

The Model v3.0 was implemented in eleven software development organizations. This paper provided details of the multiple case study included eight cases. In these cases, we observed that the organizations have challenges on the following issues: Establishing effective communication channels when there is no on-site customer, having customer's commitment, achieving an optimum level for the granularity levels of user stories, enabling the growth of product backlog at a constant pace so that development flow can be maintained, ineffective retrospective and review meetings, ability to manage technical debt, identification of the dependencies among design elements for change management.

In order to evaluate the researcher impact on the assessments, it was also implemented in three different software development organizations in the guidance of another researcher who was not involved in the development of the Model. This case study had also high success rates in terms of finding the agility issues and providing improvement suggestions. More importantly these improvement suggestions were provided using the Model by the assessment team members who were established within the software companies and had no information prior to the implementation of the Model.

In this paper, we have not proposed an assessment method describing the selection procedures of the assessed projects/products, the structure of the appraisal team or the coverage criteria of the organizational units. Such an assessment approach is saved for future work.

Another future work would be observation of the projects assessed to see if provided improvement suggestions lead to changes in organizations in terms of adopting agile software development. The current

structure of AgilityMod provides guidance in assessing the agility of software projects rather than the organizational agility. The future versions of the Model can be extended to provide guidance in assessing and achieving organizational agility, as organizational transformation dynamics would vary from transformations at project scale.

Acknowledgement

This research has been partially supported by Scientific and Technological Research Council of Turkey (TÜBİTAK), grant number 113E528 and has also been partially supported by the Science Foundation Ireland under a co-funding initiative by the Irish Government and European Regional Development Fund through Lero - the Irish Software Research Centre (http://www.lero.ie) grants 10/CE/I1855 & 13/RC/2094.

We would like to thank Onat Ege Adalı for conducting the additional three case studies which ensured the reliability of the Model.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.csi.2018.07.002.

References

[1] Agile Manifesto, 2001. Available: www.agilemanifesto.org.
[2] K. Beck, Embracing change with extreme programming, Computer 32 (10) (1999) 70–77.
[3] K. Schwaber, Scrum development process, Business Object Design and Implementation, Springer, 1997, pp. 117–134.
[4] S.R. Palmer, M. Felsing, A Practical Guide to Feature-Driven Development, Pearson Education, 2001.
[5] A. Cockburn, Crystal clear: A Human-Powered Methodology For Small Teams, Addison-Wesley Professional, 2004.
[6] K. Schwaber, Agile Project Management With Scrum, Microsoft press Redmond, 2004.
[7] D. Turk, R. France, and B. Rumpe, Assumptions Underlying Agile Software Development Processes, arXiv:1409.6610, 2014.
[8] A. Sidky, A Structured Approach to Adopting Agile Practices: The agile Adoption Framework, Virginia Polytechnic Institute and State University, 2007.
[9] M. Laanti, O. Salo, P. Abrahamsson, Agile methods rapidly replacing traditional methods at Nokia: a survey of opinions on agile transformation, Inf. Softw. Technol. 53 (3) (2011) 276–290.
[10] A. Elssamadisy, Agile Adoption Patterns: A Roadmap to Organizational Success, Addison-Wesley Professional, 2008.
[11] Ö.Özcan Top, Agilitymod: a software agility reference model for agility assessment, PhD, Information Systems, Middle East Technical University, Ankara, 20142550702031401099.
[12] ISO/IEC 15504-2:2003 Information Technology – Process Assessment – Part 2: Performing an Assessment, 2003.
[13] ISO/IEC 15504-5:2012 Information Technology – Process Assessment – Part 5: An Exemplar Software Life Cycle Process Assessment Model, 2012.
[14] Capability Maturity Model Integration (CMMI) for Development, Version 1.3, CMMI Product Team, CMU/SEI-2010-TR-033, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2010, p. 468.
[15] Ö. Özcan-Top, F. McCaffery, A hybrid assessment approach for medical device software development companies, J. Softw.: Evol. Process (2017).
[16] B. Boehm, R. Turner, Balancing Agility and Discipline: Evaluating and Integrating Agile and Plan-Driven Methods, IEEE, 2004, pp. 718–719.
[17] K. Conboy, B. Fitzgerald, Toward a conceptual framework of agile methods: a study of agility in different disciplines, Proceedings of the 2004 ACM Workshop on Interdisciplinary Software Engineering Research, ACM, 2004, pp. 37–44.
[18] J. Erickson, K. Lyytinen, K. Siau, Agile modeling, agile software development, and extreme programming: the state of research, J. Database Manag. 16 (4) (2005) 88–100.
[19] S.W. Ambler, M. Lines, Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise, IBM Press, 2012.
[20] VersionOne, The 12th Annual State of Agile™ Report, Available: http://stateofagile.versionone.com.
[21] VersionOne. (2017, 24/04/2017). The 11th State of Agiletm Report. Available: http://stateofagile.versionone.com/.
[22] S. Ambler. (2013, 24/04/2017). IT Project Success Rates Survey Results. Available: http://www.ambysoft.com/surveys/success2013.html.
[23] VersionOne. (2016, 24/04/2017). The 10th Annual State of Agiletm Report. Available: https://explore.versionone.com/state-of-agile/versionone-10th-annual-state-of-agile-report-2.
[24] Ö.Özcan Top, O. Demirörs, Assessment of agile maturity models: a multiple case study, Software Process Improvement and Capability Determination, Bremen, Germany 349 Springer, Berlin Heidelberg, 2013, pp. 130–141.
[25] A.M. Maier, J. Moultrie, P.J. Clarkson, Assessing organizational capabilities: reviewing and guiding the development of maturity grids, IEEE Trans. Eng. Manag. 59 (1) (2012) 138–159.
[26] B. Aysolmaz, O. Demirörs, A detailed software process improvement methodology: BG-SPI, European Conference on Software Process Improvement, Springer, 2011, pp. 97–108.
[27] A. Tarhan, O. Demirors, Apply quantitative management now, IEEE Softw. 29 (3) (2012) 77–85.
[28] C. Patel, M. Ramachandran, Agile Maturity Model (AMM): a software process improvement framework for agile software development practices, Int. J. Softw. Eng. 2 (1) (2009) 3–28.
[29] A. Yin, S. Figueiredo, M. Mira da Silva, Scrum Maturity Model: validation for IT organizations' roadmap to develop software centered on the client role, ICSEA 2011, The Sixth International Conference on Software Engineering Advances, 2011, pp. 20–29.
[30] R. Benefield, Seven dimensions of agile maturity in the global enterprise: a case study, System Sciences (HICSS), 2010 43rd Hawaii International Conference on, IEEE, 2010, pp. 1–7.
[31] J. Humble and R. Russell, ``The Agile Maturity Model Applied to Building and Releasing Software.
[32] N. Malic, Simple Life Cycle Agile Maturity Model, ed.
[33] M. Proulx, Yet Another Agile Maturity Model (AMM)– The 5 Levels of Maturity, (2010) Available: http://analytical-mind.com/2010/07/12/yet-another-agile-maturity-model-the-5-levels-of-maturity/.
[34] S. Jayaraj, The Agile Maturity Model, (2007) Available: http://whattodowearelikethatonly.blogspot.com/2008/08/agile-maturity-model.html.
[35] D. Leffingwell, SAFe® 4.0 Reference Guide: Scaled Agile Framework® For Lean Software and Systems Engineering, Addison-Wesley Professional, 2016.
[36] I. Stojanov, O. Turetken, J.J. Trienekens, A maturity model for scaling agile development, Software Engineering and Advanced Applications (SEAA), 2015 41st Euromicro Conference on, IEEE, 2015, pp. 446–453.
[37] R.M. Fontana, V. Meyer, S. Reinehr, A. Malucelli, Progressive outcomes: a framework for maturing in agile software development, J. Syst. Softw. 102 (2015) 88–108.
[38] R.M. Fontana, S. Reinehr, A. Malucelli, Agile compass: a tool for identifying maturity in agile software-development teams, IEEE Softw. 32 (6) (2015) 20–23.
[39] O.E. Adalı, Ö. Özcan-Top, O. Demirörs, Evaluation of agility assessment tools: a multiple case study, International Conference on Software Process Improvement and Capability Determination, Springer, 2016, pp. 135–149.
[40] O.R. Yürüm, O. Demirörs, Agile maturity self-assessment surveys: a case study, Software Engineering and Advanced Applications (SEAA), 2017 43rd Euromicro Conference on, IEEE, 2017, pp. 392–399.
[41] T. Schweigert, D. Vohwinkel, M. Korsaa, R. Nevalainen, M. Biro, Agile Maturity model: a synopsis as a first step to synthesis, Systems, Software and Services Process Improvement, Springer, 2013, pp. 214–227.
[42] T. Schweigert, D. Vohwinkel, M. Korsaa, R. Nevalainen, M. Biro, Agile maturity model: analysing agile maturity characteristics from the SPICE perspective, J. Softw.: Evol. Process (2013).
[43] T. Schweigert, R. Nevalainen, D. Vohwinkel, M. Korsaa, M. Biro, Agile maturity model: oxymoron or the next level of understanding, Software Process Improvement and Capability Determination, Springer, 2012, pp. 289–294.
[44] F. McCaffery, M. Pikkarainen, I. Richardson, Ahaa–agile, hybrid assessment method for automotive, safety critical smes, Proceedings of the 30th international Conference on Software Engineering, ACM, 2008, pp. 551–560.
[45] C. Bianco, Agile and SPICE capability levels, Software Process Improvement and Capability Determination, Springer, 2011, pp. 181–185.
[46] A. Omran, AGILE CMMI from SMEs perspective, Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on, IEEE, 2008, pp. 1–8.
[47] R. Turner, A. Jain, Agile meets CMMI: culture clash or common cause? Extreme Programming and Agile Methods—XP/Agile Universe 2002, 2002, pp. 153–165.
[48] T. Kovacheva, N. Todorov, Optimizing software development process: a case study for integrated Agile-CMMI process model, EUROCON-International Conference on Computer as a Tool (EUROCON), IEEE, 2011, pp. 1–2.
[49] J. Sutherland, C.R. Jakobsen, K. Johnson, Scrum and CMMI level 5: the magic potion for code warriors, Hawaii International Conference on System Sciences, Proceedings of the 41st Annual, IEEE, 2008466-466.
[50] J. Wäyrynen, M. Bodén, G. Boström, Security engineering and eXtreme programming: an impossible marriage, Conference on Extreme Programming and Agile Methods, Springer, 2004, pp. 117–128.
[51] F.S. Silva, et al., Using CMMI together with agile software development: a systematic review, Inf. Softw. Technol. 58 (2015) 20–43.
[52] C. Torrecilla-Salinas, J. Sedeño, M. Escalona, M. Mejías, Agile, web engineering and capability maturity model integration: a systematic literature review, Inf. Softw. Technol. 71 (2016) 92–107.
[53] P.E. McMahon, Integrating CMMI and Agile Development: Case Studies and Proven Techniques For Faster Performance Improvement, Addison-Wesley Professional, 2010.
[54] Ö. Özcan-Top and O. Demirörs, CMMI ve Çevik Yazılım Geliştirme Yöntemlerinin Birlikte Uygulanabilirliği, in Ulusal Yazılım Mühendisliği Sempozyumu, 2013: CEUR Workshop Proceedings, İzmir, Türkiye.
[55] J. Becker, R. Knackstedt, J. Pöppelbuß, Developing maturity models for IT management, Bus. Inf. Syst. Eng. 1 (3) (2009) 213–222.
[56] A. Hevner, S. Chatterjee, Design science research in information systems, Design Research in Information Systems, Springer, 2010, pp. 9–22.

[57] J.W. Creswell, Research design: Qualitative, Quantitative, and Mixed Methods Approaches, Sage Publications, Inc, 2009.
[58] T. Dybå, R. Prikladnicki, K. Rönkkö, C. Seaman, J. Sillito, Qualitative research in software engineering, Empir. Softw. Eng. 16 (4) (2011) 425–429.
[59] R.K. Yin, Case Study Research: Design and Methods, Sage publications, 2014.
[60] M. Rohloff, Case study and maturity model for business process management implementation, International Conference on Business Process Management, Springer, 2009, pp. 128–142.
[61] C. Robson, Real World Research, 2nd Edition, Blackwell Publishing, Malden, 2002.
[62] Ö.Özcan Top, O. Demirors, Assessing software agility: an exploratory case study, Software Process Improvement and Capability Determination 477 Springer, Vilnius, 2014, pp. 202–213.
[63] M. Salmanoğlu, A. Coşkunçay, A. YILDIZ, O. Demirörs, An exploratory case study for assessing the measurement capability of an agile organization, Softw. Qual. Prof. 20 (2) (2018).
[64] A. Uskarcı, O. Demirörs, Do staged maturity models result in organization-wide continuous process improvement? Insight from employees, Comput. Stand. Interfaces 52 (2017) 25–40.
[65] A. Uskarcı, O. Demirörs, A case study on employee perceptions of organization wide continuous process improvement activities, International Conference on Software Process Improvement and Capability Determination, Springer, 2012, pp. 26–37.
[66] Ö. Özcan Top, AgilityMod: Software Agility Assessment Reference Model v3.0, Informatics Institute, METU/II-TR-2014-392014.
[67] K.S. Rubin, Essential Scrum: A Practical Guide to the Most Popular Agile Process, Addison-Wesley Professional, 2012.
[68] Ö.Özcan Top, AgilityMod: Software Agility Assessment Reference Model v3.0 Application: Case Study Results, Informatics Institute, 2014 METU/II-TR-2014-40.
[69] O.E. Adalı, Assess Agility: Agility Assessment Approach Supported With an Automated Web Based Agility Assessment Tool, MSc., Information Systems, Middle East Technical University, 20172550702031700218.
[70] O.E. Adali, Ö.Ö. Top, O. Demirörs, Assessment of agility in software organizations with a web-based agility assessment tool, Software Engineering and Advanced Applications (SEAA), 2017 43rd Euromicro Conference on, IEEE, 2017, pp. 88–95.

Özden Özcan-Top is a post-doctoral researcher at the Regulated Software Research Centre, Dundalk Institute of Technology, Ireland. She received her Master's and PhD degrees from Middle East Technical University, Informatics Institute, Information Systems program, Turkey. She has been working as a researcher in the areas of software project management, software process improvement, software measurement and benchmarking since 2007. She worked as a quality specialist at a private software development company and managed the CMMI-Dev based process improvement program for 2,5 years. Her current research interests focus on medical device software development standards and adoption of agile software development methods in highly regulated environments.

Onur Demirörs has joined University of New South Wales as a visiting professor. He is a professor of the Computer Engineering Department at Izmir Institute of Technology and the strategy director of Bilgi Grubu Ltd. He took his Ph.D. degree in Computer Science from Southern Methodist University. His research focuses on process modeling and improvement, business process management, software measurement, software engineering standards, and organizational change management. He leaded major research projects on developing process improvement and modeling techniques for SMEs, on establishing and implementing business process modeling approaches for large scale organizations and on establishing measurement infrastructures for software organizations. He worked as a consultant for over 20 companies to improve their processes, to establish their measurement infrastructures and to create organizational knowledge structures.