Turkish Journal of Electrical Engineering and Computer Sciences

Volume 30 | Number 4 Article 3

1-1-2022

A survey on organizational choices for microservice-based software architectures

HÜSEYİN ÜNLÜ

BURAK BİLGİN

ONUR DEMİRÖRS

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Review Article

Turk J Elec Eng & Comp Sci
(2022) 30: 1187 – 1203
© TÜBİTAK
doi:10.55730/1300-0632.3843

A survey on organizational choices for microservice-based software architectures

Hüseyin ÜNLÜ[*], **Burak BİLGİN**, **Onur DEMİRÖRS**
Computer Engineering Department, İzmir Institute of Technology, İzmir, Turkey

Received: 14.09.2021 • **Accepted/Published Online:** 17.03.2022 • **Final Version:** 31.05.2022

Abstract: During the last decade, the demand for more flexible, responsive, and reliable software applications increased exponentially. The availability of internet infrastructure and new software technologies to respond to this demand led to a new generation of applications. As a result, cloud-based, distributed, independently deployable web applications working together in a microservice-based software architecture style have gained popularity. The style has been a common practice in the industry and successfully utilized by companies. Adopting this style demands software organizations to transform their culture. However, there is a lack of research studies that explores common practices for microservices. Thus, we performed a survey to explore the organizational choices on software analysis, design, size measurement, and effort estimation when working with microservices. The results provide a snapshot of the software industry that utilizes microservices. We provide insight for software organizations to transform their culture and suggest challenges researchers can focus on in the area.

Key words: Microservice architecture, software analysis, software design, software size measurement, software effort estimation, practitioner survey

1. Introduction

Today, applications can utilize virtually unbounded resources utilizing cloud platforms. This potential can be enabled by distributing business logic on platforms[1]. Microservice-based architecture is one of the most frequently explored solutions for this purpose. Microservices are loosely coupled applications that work in cohesion. They can be deployed, scaled, and tested independently and communicate over a network using lightweight asynchronous messages by interfaces like REST [1–3].

The popularity of microservice architecture has increased rapidly in the industry especially since 2015 [4]. Today's massive service providers such as Amazon, LinkedIn, Netflix, SoundCloud, Uber, and Verizon have adopted microservice-based approaches [5].

Microservices are developed and deployed independently which support essential attributes such as scalability, unfragility, and maintainability for a wide variety of modern systems. Although microservices are services, there are three main characteristics of microservice-based architecture that differ from service-oriented architecture (SOA): size, bounded context, and independence [6]. Firstly, the size of the microservice is smaller than the typical service. A microservice provides only one single business capability. Secondly, a microservice

[*]Correspondence: huseyinunlu@iyte.edu.tr

[1]Fowler M, Lewis. Microservices [online]. Website https://martinfowler.com/articles/microservices.html [accessed 11 March 2022].

combines all related functionalities into a single business capability. Thirdly, microservices are independent services. As each microservice has a single responsibility, there should be a single reason to change or replace each microservice. Thus, other microservices should not be affected by a change in another microservice.

The concept sounds quite familiar yet transforming to develop for microservice-based architectures has unique challenges. Organizations should deal with decentralization and autonomy on one hand and efficient management and integration of the output of all teams on the other. In addition, the experts in the field observe again and again that traditional modeling notations used for the analysis and design are not effective when working with microservices [7]. Even the most fundamental abstraction we use today the 'object' of the Object-oriented Analysis and Design (OOAD) methodologies might have to change. Similarly, we have observed that traditional functional size measurement methods do not fit the structure of microservices [8]. In this survey, our goal is to visualize the current practice in relation to these challenges.

In the early phases of our study, we first aimed to do a systematic literature review on organizational choices when working with microservice-based architectures and gather information about the best practices on the analysis, design, size measurement, and effort estimation of microservices. These software development life cycle (SDLC) activities play an important role in software development and developers should not treat them as after-the-fact documentation. We performed a preliminary search from following libraries to find relevant studies: (1) ACM Digital Library, (2) Google Scholar, (3) IEEE Xplore, (4) Science Direct, (5) SpringerLink. The preliminary search resulted in 43 studies but, only a few of them was related to our aim (see Section 2). Overall, we have seen that there is a lack of research studies in the literature that explores the common practices for microservices. Thus, we changed our research method to perform a survey to explore organizational choices. The main contributions of our submission can be summarised as follows: (1) With this work, we provide survey results of 67 participants in 9 countries from the field, on software analysis, design, and management when working with microservices. (2) We provide insight for software organizations to transform their culture for microservice-based software architectures. (3) We suggest a set of challenges researchers can focus on transformation for microservice-based software architectures in the form of new analysis, design, and size measurement approaches.

The remainder of this paper is structured as follows. In Section 2, we mention the related work from the literature. In Section 3, we explain our research methodology behind the survey. In Section 4, we give our results. In Section 5, we discuss our findings. Lastly, in Section 6, we conclude our study by stating the future work.

2. Related work

Only a few studies have surveyed software professionals regarding microservices. Bandeira et al. [9] analyzed microservice-related discussions on Stack Overflow. They categorized 1043 microservice-tagged posts into three: technical (18 subjects), conceptual (15 subjects), and nonrelated discussions. Their analysis shows that Netflix Eureka and Token Authentication are the most viewed and answered technical and conceptual discussions, respectively. On the other hand, Message Queue is one of the most popular discussions for both categories.

Soldani et al. [10] decomposed the pains and gains of designing, developing, and operating microservices by analyzing 51 industrial studies. They listed the pains as dimensioning of microservices and security at design-time, storage, and testing at development-time and resource consumption at operating time. On the other hand, the gains are listed as decentralization by assigning the governance of different microservices to different groups, natural enforcement of fault tolerance at design-time, the freedom in choosing the technology stack for developing a microservice, and the ease of use of DevOps techniques at development-time and the possibility of independently deploying the microservices in an application at operation-time.

Bogner et al. [11] performed 7 interviews with software professionals from 10 companies and analyzed 14 service-based systems to identify the technologies, characteristics, and software quality-related issues on microservices. Their interviews show that RESTful HTTP is the de facto standard and Docker containers are used for the deployment of services in general. In the majority of the systems, operations are done by different people. The technological heterogeneity level differs between the projects. Service granularity was also very heterogeneous. Their results also show that maintainability is the most obvious improvement over the monolith.

Ghofrani and Lübke [12] surveyed the challenges of microservices architecture in practice. Their survey is based on three research questions regarding the main challenges in the design and the development process, main reasons for leveraging and preventing the usage of systematic approaches in microservices architectures, and suggestions from the experts to improve aspects of the microservices architectures. According to the 25 collected results, there are three main challenges: the distributed nature of microservices architecture, skill and knowledge, correct separation of domains, and finding the appropriate service cuts. The participants mention that security, performance, and response time as very important to optimize in microservices. Also, most participants set their goals as scalability and agility when deciding on a microservices architecture. The majority of the participants do not use any notations to describe the architecture. However, graphical modeling languages are the most used notation among participants. Lastly, if they use any artifacts from third parties in their microservice project, security, license, and memory usage are the most respected concerns. They aim to conduct the survey each year to monitor the change over time.

Zhou et al. [13] conducted an industrial survey to reveal typical faults of microservice systems, the current practice of debugging, and challenges practice. They interviewed 16 participants from 12 companies. Their interview resulted in 22 typical microservice fault cases which include functional and nonfunctional influences on microservice systems. They listed the root causes such as the internal implementation of individual microservices, interactions among multiple microservices, or the configuration of the runtime infrastructure. Their survey result shows that developers heavily depend on log analysis for fault analysis and debugging. They categorized existing practices and techniques on debugging into three maturity levels such as basic log analysis, visual log analysis, and visual trace analysis. After the survey, the authors replicated the 22 fault cases to develop a benchmark microservice system to investigate the effectiveness of existing industrial debugging practices and propose possible improvements. They found that trace visualization can be beneficial for most fault cases (especially the ones related to microservice interactions). They also found that microservice debugging can be improved using debugging visualization tools for distributed systems from the literature.

Taibi et al. [14] surveyed practitioners to identify the common motivations and issues during migration from monolith to microservices. Their results show that maintainability and scalability are the most common motivations while the delegation of team responsibilities, DevOps support, fault tolerance, easy technology experimentation, the delegation of software responsibilities, and the common sense for migration was also reported as important motivations. The main migration issues were reported as complexity to decouple from the monolithic system, splitting of data in legacy databases, and communication among services alongside effort estimation, DevOps infrastructure effort, library conversion effort, people's mind, and expected long-term return on investment.

Carvalho et al. [15] performed a survey to assess what criteria are useful to extract microservices during the migration to microservice architecture. Their results based on 15 participants show that academic techniques do not totally satisfy the needs of practitioners. For the majority of the participants, coupling, cohesion, and reuse are the most dominant criteria. However, they found that existing tooling support is insufficient or irrelevant to support their microservice extraction decision.

Di Francesco et al. [16] surveyed 18 participants to understand the performed activities and the challenges faced during the migration. The main challenges are reported as the high level of coupling, the difficulties in identifying the boundaries of services, and system decomposition.

Balalaie et al. [17] presented 15 migration and rearchitecting patterns that are identified from industrial software migration projects. They also proposed a methodology based on small and medium-sized entrepreneurs (SMEs) for the selection and composition of the migration patterns to prepare a migration plan. Their observation shows that there is a still need for improvement of the pattern's application and their methodologies.

Viggiato et al. [18] reported the results of a survey with 122 professionals who work with microservices. Their results show that four programming languages are largely used: Java (33%), JavaScript through Node.js (18%), C# (12%), and PHP (8%). The most common database management system (DBMS) is Postgres. Sixty-two percent of the participants declared that they use REST over HTTP. The participants found that the independent deployment is the most important advantage of microservices while ease to scale application, maintainability and no commitment to single technology follows this advantage. The complex distributed transactions were found as the most important challenge were testing the whole system, service faults and expensive remote calls follow this challenge respectively.

Vural et al. [19] performed a systematic literature review on microservices to explore the research type, the main practical motivations behind microservices related research, and the emerging standards and de facto tools on microservices solutions. Their results show that the research is around solution proposals that do not explicitly call out the possible service types applicable for that solution. As the second result, the main motives are around functionality followed by performance and test techniques. As the third result, REST can be called out as the standard for Microservices. Also, Docker is seen as the most frequently used tool in studies followed by Node.js, MySQL, Postgres, and Java.

The studies from the literature on microservices mainly focus on challenges in migrating to microservices. Some studies include limited information about the analysis and design of microservices, however, they do not reveal the practices in terms of methodologies and modeling notations used for analysis and design tasks. No study reveals the organizational choices on size measurement and effort estimation of microservice-based projects. Therefore, we observed the need for surveys that reveals the management aspects of microservice-based projects. Accordingly, our survey focuses on three main perspectives: analysis, design, size measurement, and effort estimation.

3. Research methodology

We have utilized a survey methodology to gain insight into how companies cope with microservice-based architectures. As for the way we conduct our survey; we chose to do an anonymous online survey for three reasons. Firstly, we wanted to reach out to a large number of organizations across the world quickly. As for our choice of anonymity, we believed that companies and their employees would be hesitant on sharing their information on the development life cycle. Lastly, we wanted the data collected to be easy to categorize and analyze. As for the platform, we chose Google Forms as our platform for the survey as it is easy and free to use, can export results in XLSX format easily, and represents data collected in an informative manner.

One of the main problems a survey can have is ambiguous questions [20]. To check for these kinds of questions, we also conducted a pilot experiment when the survey was prepared. This pilot experiment was used to observe survey completion time by someone who saw the survey for the first time and also to gain feedback on the questions. After the pilot was conducted, some corrections were made, and the survey was executed.

3.1. Goal and research questions

The main goal of this survey is to learn which practices companies use when they work with microservice-based architectures. Based on the results, we aim to detect which practices are popular, which are not, and the shortcomings of both. With these goals in mind and the scope of this article, the research questions (RQs) are as follows:

- Which analysis techniques do companies utilize when working with microservice-based architectures?
- Which design patterns do companies utilize when working with microservice-based architectures?
- Which size measurement and effort estimation techniques do companies utilize when working with microservice-based architectures?

3.2. Sampling method

As for our sampling method, we used accidental nonprobabilistic sampling [21]. Our target group of participants was professionals in the software field with experience in microservice-based architectures. To find such individuals, we mainly used our own personal or company contacts, forums, mail groups, and LinkedIn for finding and contacting our participants. We also asked our participants to share our survey with their colleagues and their own contacts if the number of participants of the same company does not pass five persons. As the survey got more distributed, we had no way of tracking who participated in our survey due to its anonymity. However, we divided our participants on whether they have experience with microservice-based architectures or not to analyze the data gathered.

3.3. Designing survey questions

As the number of our questions increased, we deemed it necessary to divide our survey into several sections. The final version of our survey consisted of six sections. The first section acts as an overview of the survey and the participant's approval of content. The second section served as our demographic section and aims to collect general information on participants and their organization, their general experience, and detect whether they have experience with microservices. Also, if the participant chooses that they do not have any experience with microservices, this section ends the survey prematurely. In the third section, we aimed to obtain general data on participants' experiences with microservices, such as the type of their last microservice-based project (new development, reimplementation of a monolith, etc.), the tools they use, the critical challenges they encountered. The fourth section focuses on the size measurement and effort estimation techniques they used while working with microservice-based architectures. In the fifth section, we focused on questions related to software analysis techniques, and in the sixth section, we focused questions on software design patterns they followed while working with microservices. The list of questions is given in Table 1.

Table. List of the questions developed and used in the survey.

Section	Survey question	Type of answers			
		Single answer from a list	Multiple answers from a list	Free text field	Likert scale
2	What is the origin of your current organization?			X	
	What is your formal undergraduate education?	X			
	What is your latest formal graduate education?	X			
	What is your role in your organization?	X			
	How long (in years) have you been working in your current role?			X	
	How long (in years) have you been working in the software field?			X	
	Do you have any experience with microservice-based architectures?	X			
3	How long (in years) have you been working with microservices?			X	
	What is the domain of your current (if finished, the last) microservice-based project?	X			
	What is the type of your current (if finished, the last) microservice-based project?	X			
	Which software development methodology you followed when working with microservices if you followed any?		X		
	What is the team size (total number of personnel) on your current (if finished, the last) microservice-based project?			X	
	How many microservice-based projects are running concurrently in your organization?			X	
	Do developers in your organization generally work on one project at a time or do they work on multiple projects at a time?	X			
	Do you utilize DevOps in your organization in general?	X			
	Which automated integration and automated deployment tools you use if you use any?			X	
	What is the level of test automation in your current (if finished, the last) microservice-based project?				X
	What was the most critical challenge you experienced in your software development process when working with microservices in your opinion?	X			
4	Which software size measurement method have you used in your current (if finished, the last) microservice-based project if you used any?	X			
	How do you perform effort estimation in your current (if finished, the last) microservice-based project?	X			

Table. Continued

	Which tasks are included in effort estimation in your current (if finished, the last) microservice-based project?		X		
	Have you been able to predict effort more precisely when utilizing microservice architectures?				X
4	Have you observed changes in the efficiency of the effort estimation process when utilizing microservices architectures?				X
	How frequently is actual individual effort recorded in your organization?			X	
	Does your organization utilize a standard process for software analysis in general?	X			
5	How have you depicted the functional requirements in your current (if finished, the last) microservice-based project?		X		
	How have you analyzed the problem for your current (if finished, the last) microservice-based project?		X		
	Which notations have you used to analyze your current (if finished, the last) microservice-based project?		X		
	How have you decomposed the problem into microservices in your current (if finished, the last) microservice-based project?	X			
6	Which notations have you used to represent the design of your current (if finished, the last) microservice-based project?		X		
	Which patterns have you utilized in your current (if finished, the last) microservice-based project?		X		

3.4. Survey piloting and execution

To check against the ambiguity of the questions and prevent misunderstandings, a pilot experiment was conducted with 3 individuals in different companies and different roles. Based on the feedback received, the time to complete the survey was calculated and the changes below were made.

- The survey required logging in with a Gmail account, this setting was disabled.
- A control question was added for checking whether the participant has any experience with microservices and ending the survey based on that.
- Questions that were aimed to learn about the experience of the participant were changed from monthly to yearly intervals.
- Some questions' answers were changed from free-text areas to multiple choices with the 'Other:' option.

After the pilot was completed, the last changes in the survey were made. After changes, the survey was sent to the Human Subjects Ethics Committee (HSEC) of İzmir Institute of Technology (IZTECH) for approval.

3.5. Criteria for validation

As for our criteria of validation, we relied mainly on the seventh question (Q7) which asks whether the participant has experience in microservice-based architecture or not. Even though the word 'microservice' is emphasized in the title of the survey and the invitation, we sent to our participants, there were still some participants who filled the survey that did not have any experience with microservices. The last question in Section 2 was used to check for this circumstance and finish the survey early if it was answered 'No'. Also, after the execution of the survey is complete, the group that has no experience will be excluded from the analysis.

4. Survey results

In this section, due to space constraints, we report a summary of the survey results of 67 participants. Our raw data can be found at https://bit.ly/305cuiC.

4.1. Demographics of the participants

The origin of the 67 participants includes 9 countries from 4 continents: Turkey, Australia, Germany, Italy, China, Netherlands, Southern Cyprus, the United Kingdom, and the USA (see Figure 1a). The majority of our participants (85%) are from Europe while 12% is from America, 1.5% is from Asia and 1.5% is from Australia. The formal undergraduate education of the participants includes Computer Engineering, Computer Science, Software Engineering, Information Systems, Electrical and Electronics Engineering, Industrial Engineering, and other disciplines such as Mechanical/Mechatronics Engineering, Mathematics Engineering, Business and Economics, and Industrial Engineering (see Figure 1b). Majority of the participants graduated from Computer Engineering (72%) while 6% graduated from Computer Science and 5% graduated from Electrical and Electronics Engineering. Almost 73% of the participants have a graduate degree. The graduate studies of the participants include Computer Engineering, Computer Science, Information Systems, Software Engineering, and other disciplines such as Business, Cyber Security, Sustainable Development, Electrical and Electronics Engineering, Industrial Engineering, and Management Science (see Figure 1c). Similar to an undergraduate degree, the majority of the participants' graduate degrees is from Computer Engineering (42%). Ten percent graduated from Computer Science, 6% graduated from Information Systems and 5% graduated from Software Engineering. The participants have a variety of roles in their respective organizations, which are Developer, Senior Developer, Software Architect, Project Manager, Analyst, and other roles such as CTO, MSc. Student, Research Assistant, Software Test Engineer, and System and Network Consultant (see Figure 1d). Most participants work as a developer (46%) or senior developer (19%). Ten percent works as a software architect, 10% works as a project manager and 3% works as an analyst.

The experience of the participants in their current roles ranged from 0.2 years to 15 years with an average of 2.95 years (see Figure 1e). The median experience is 2, the mode is 1 and the standard deviation is 3.10. The majority of the participants have experience in their current role of fewer than 2.7 years. Their total experience in the software field ranged from 0.2 to 30 years with an average of 6.50 years (see Figure 1f). The median is 3, the mode is 2 and the standard deviation is 6.61. The majority of the participants have experience in their current role of fewer than 5.7 years.

We divided our participants on whether they have experience with microservice-based architectures or not to analyze the data gathered. A question at the end of this part was used for this purpose. After the execution of the survey was complete, the group that has no experience was excluded from the analysis. Of the

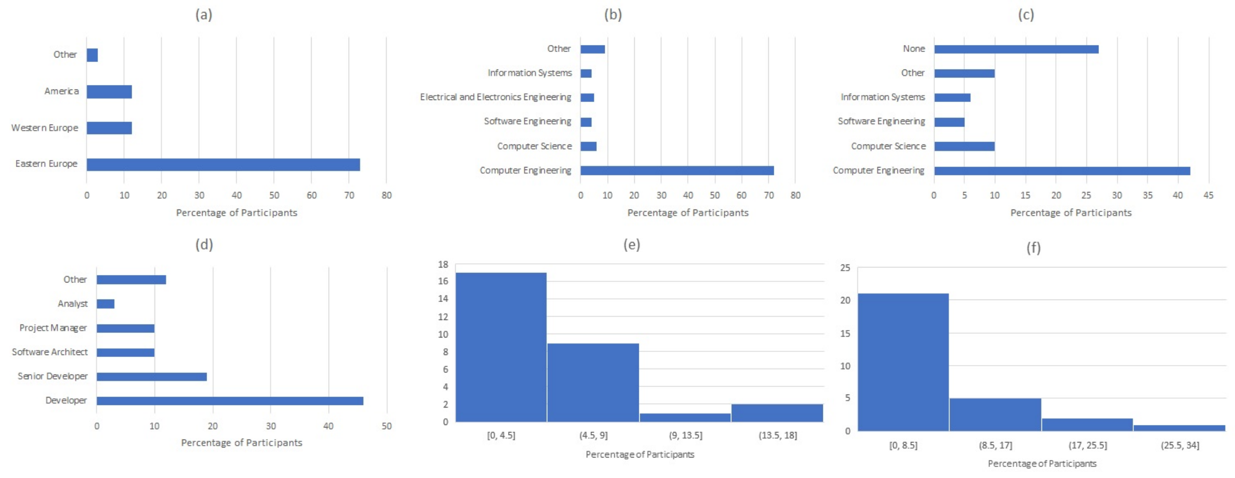

Figure 1. Participants responses regarding their (a) organizations' geographical distribution, (b) undergraduate studies, (c) graduate studies, (d) role in the organization, (e) experience in the current role (in years), and (f) experience in the software field (in years).

67 people who participated in the survey, 45 participants (67%) had experience with microservices.

4.2. Experience with microservice-based projects

The average experience of the participants with microservices is 2.4 years. The median experience is 2, the mode is 1 and the standard deviation is 1.9. The majority of the participants have experience in their current role of fewer than 2.2 years. The last microservice-based projects of the participants represent 20 domains. The majority of the represented domains are finance, telecom, mobile software, web-based productivity apps, e-commerce, and entertainment.

When we inspect their last microservice-based project; 47% developed a new microservice-based system, 36% reimplemented a monolithic system from scratch as microservices, 16% reengineered parts of a system as microservices to work with a monolith, and 2% replatformed an already microservice-based system.

The majority of the participants follow Scrum as a software development methodology (53%). Kanban is the second popular methodology among the participants (24%). Nine percent of the participants follow Extreme programming.

The average team size on the last microservice-based project is 9.3. The median team size is 5, the mode is 5 and the standard deviation is 9.43. Most teams include less than 11.3 members. The average number of a microservice-based project concurrently running in the organizations is 40.11. The median number of projects is 5, the mode is 1 and the standard deviation is 194.96. Most organizations are concurrently running less than 190 microservice-based projects.

Almost 60% of the participants generally work on multiple projects while the remaining 40% work on one project at a time.

Continuous Integration, which is the core concept of Continuous Software development, is a set of software engineering practices that speed up the delivery of software by decreasing integration times [22]. DevOps is a set of practices that combines software development and IT operations to provide continuous delivery with high software quality. In 84% of the participated organizations, DevOps is utilized in general. Jenkins is the most preferred automated integration and deployment tool among the participants. Thirty-two percent of the participants use Jenkins[2], 12% use Azure DevOps[3] and 7% do not use any tool. In their studies, Bogner et al. [11] and Vural et al. [19] reported that Docker[4] containers are mostly preferred for deployment among their participants. Our results are not in parallel with these studies. Although Docker container is used by some of our participants it is not mostly the preferred deployment tool.

When we look at the level of test automation in their last microservice-based project, the majority voted as automated from the Likert scale (37%). Also, more than 50% of the participants voted automated or highly automated.

We also asked about the most critical challenge they face when working with microservices. Participants reported two main challenges: the complexity of decomposing a monolithic system (36%) and managing transactional processes where multiple services need to commit/roll back changes together (36%). Sixteen percent reported managing asynchronous communication among services and 7% reported database migration and data splitting as the most critical challenge. Our results about the most critical challenge are in parallel with the results of the studies in the literature [10, 12, 14, 16, 18].

4.3. Analysis and design

Microservice-based architecture differs from traditional architectures in several ways, even the most fundamental abstraction the ‘object’ might have to change as in most cases coupling through objects impacts autonomy in a negative manner. The analysis and design approach should provide a natural decomposition strategy for developing a microservice-based solution considering bounded contexts and asynchronous communication. But how do organizations perform analysis and design of microservice-based projects?

Sixty percent of the participants utilize a standard process for software analysis for microservice-based projects (see Figure 2a). They depict the functional requirements based on traditional approaches (see Figure 2b). Sixty-nine percent of the participants depict the functional requirements using user stories while 36% utilize use case scenarios and 29% utilize unstructured/natural language. Only 1 participant uses an event-based notation which is event-driven process chain (EPC) to depict the functional requirements. We observe that although the decomposition strategy of microservice-based architecture should not be based on classes, object-oriented analysis approaches are commonly applied to depict the functional requirements for microservice-based projects.

In the problem analysis stage, organizations mainly follow event-based and ad-hoc analysis techniques (see Figure 2c). Almost 36% of the participants use event-storming to analyze the problem for microservice-based projects. Thirty-three percent apply ad-hoc analysis and 31% apply event-based modeling. On the other hand, 29% of the participants do not use any technique to analyze the problem.

For the notations they use to analyze the microservice-based projects, we expected that the participants use event-based notations as they mainly analyze the problem based on event-based techniques. However, the results are surprising, the majority use the same modeling notations as they use in more conventional projects (see Figure 2d). Almost 53% of the participants use flow charts and 38% use activity diagrams. Similar to problem analysis, 31% do not use any notation. Flow charts and activity diagrams may help to decompose the problem by bounded context as they represent the flow from one activity to another activity. Service in a

[2]Jenkins [online]. Website https://www.jenkins.io/ [accessed 06 January 2022].
[3]Azure DevOps [online]. Website https://azure.microsoft.com/en-us/services/devops/ [accessed 06 January 2022].
[4]Docker [online]. Website https://docker.com [accessed 06 January 2022].

Microservice Architecture is best to be triggered by an event [8]. Thus, event-based notations such as EPC are expected to be used more frequently, yet it is used by only 4% of the participants.

The majority of the participants decompose the problem into microservices by bounded context (see Figure 2e). Approximately 73% of the participants decompose by bounded context while the remaining 27% decompose by business capability.

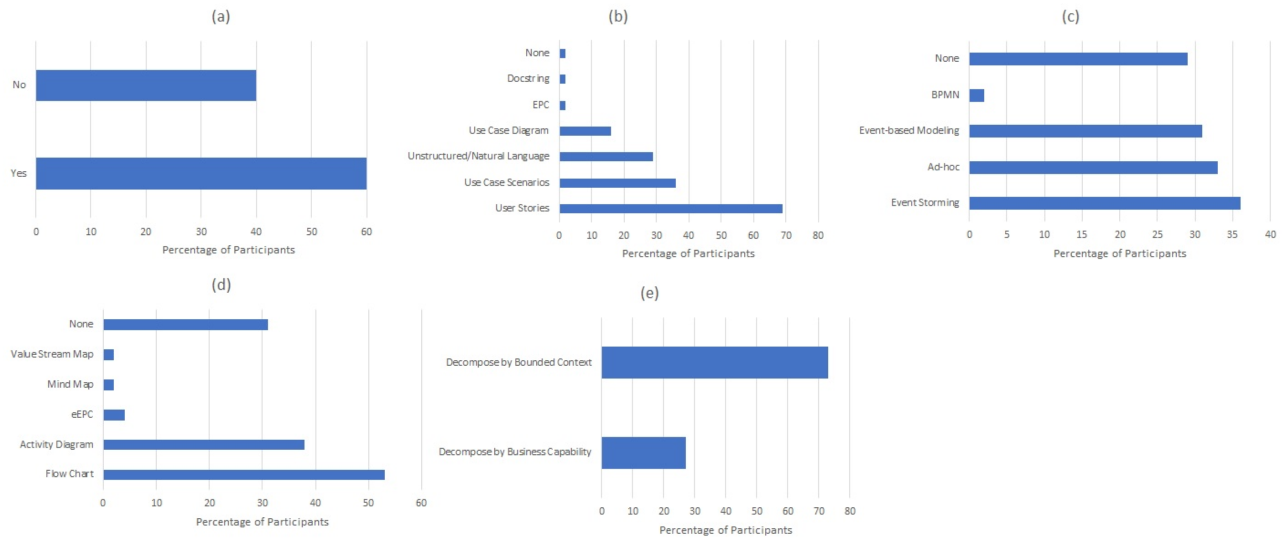

Figure 2. Participants' responses regarding (a) utilizing a standard process for software analysis, (b) techniques to depict functional requirements, (c) techniques to analyze the problem, (d) notations to analyze the project, and (e) decomposition of the problem into microservices.

The software design process needs decisions on the software system. However, even in traditional architectures, developers often face uncertainty about making such decisions [23]. In contrast, design-time uncertainty cannot be ignored as software modeling tools, languages, and techniques are based on an input that does not include any uncertainty. This uncertainty may become more challenging with new paradigms such as microservices.

The software design part of the survey covers two questions: notations to represent the design and utilized patterns in microservice-based projects. Most organizations prefer to use conventional UML diagrams to represent the software architecture of traditional systems [24]. However, the design of microservice-based projects is different from traditional systems as the context is not (or should not be) bounded by the objects.

Surprisingly, most participants use modeling notations most suitable for object-oriented analysis and design for their microservice-based projects (see Figure 3a). Almost 58% of the participants use sequence diagrams and 51% use class diagrams. Other answers include two other notations: the activity diagram (38%) and ER diagram (27%). Also, 18% of the participants do not use any design notation to describe the architecture which is a parallel result with the findings of Ghofrani and Lübke [12]; they reported that the majority of the participants do not use any modeling notations to describe the architecture. In another study, Alshuqayran et al. [25] mentioned that component/container, process/behavior, sequence, execution timeline, deployment, class, and use case diagrams are the most frequently used diagrams in the literature for microservice-based projects. The results of our survey are quite similar although somewhat surprising.

In a microservice-based architecture, several patterns are used to provide system reliability. If any service fails, the rest of the system should continue to operate. Our result shows that event sourcing and circuit breaker is the most utilized patterns in microservice-based projects (see Figure 3b). Almost 58% of the participants utilize event sourcing, 51% utilize circuit breaker and 36% utilize command query responsibility segregation (CQRS).

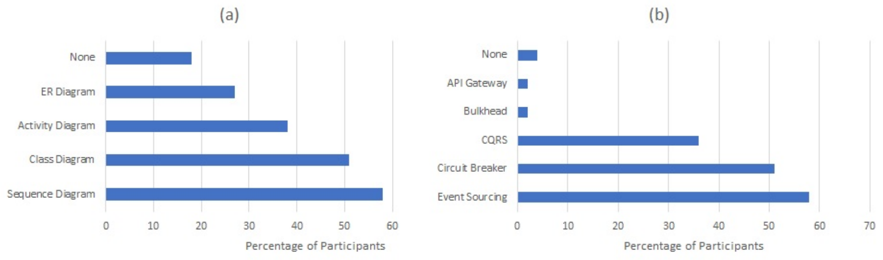

Figure 3. Participants' responses regarding (a) notations to represent design and (b) utilized patterns.

In summary, our survey showed that most of the organizations follow a process for the analysis and design of microservice-based projects. However, there is no de facto methodology as developing solutions using object-oriented analysis and design approaches. Organizations use a combination of different approaches. Most organizations still utilize object-oriented analysis and design (OOAD) notations to develop microservice-based projects although these notations might not be most helpful to decompose systems into microservices.

4.4. Size measurement and effort estimation

Software size measurement and effort estimation are significant activities in software management and provide significant business advantages [26]. Especially, successful effort estimation may reduce potential risks on schedule and cost such as schedule and budget overrun. Most of the size measures are built on conventional design entities of related analysis and design approaches such as function points (based on the transaction concept [27]), object points (based on the object concept), story points (based on the user story concept). As these entities are changing in Microservice architectures, we wondered how do organizations respond to the challenge.

Most participants use story points as software size measurement methods in microservice-based projects (38%). Eighteen percent use T-shirt size, 9% use source lines of code, and 9% use use-case points. Twenty-two percent of the participants do not use any size measurement method (see Figure 4a). In the majority of organizations, effort estimation is performed based on planning poker (31%). Twenty-four percent apply expert judgment, 16% apply ad-hoc and 11% apply parametric application (see Figure 4b).

Effort estimation includes several tasks such as development, analysis, design, test, and operation. Development effort is included in 91% of the organizations. Test effort is also included in most of the organizations (76%). Fifty-eight percent of the participants include analysis effort, 51% include design effort and 38% include operations effort in their organizations. In 11% of the organizations, the effort is not estimated (see Figure 4c).

The majority of the participants agree that the effort is predicted more precisely when utilizing microservice architectures (see Figure 4d) while 9% of the participants strongly agree (voted as 5), 42% of the participants agree (voted as 4) with the statement. Twenty-nine percent of the participants observe that microservice architectures do not affect the precision (voted as 3).

Fifty-one percent of the participants do not see any changes (voted as 3) in the efficiency of the effort estimation process when utilizing microservice architectures (see Figure 4e) while 24% find it efficient (voted as 4) and 13% find it very efficient (voted as 5). Most participants record actual individual effort daily (31%) or weekly (37%). Twenty-two percent of the participants do not record individual effort (see Figure 4f). This percentage is the same as the percentage of participants who do not use any size measurement method.

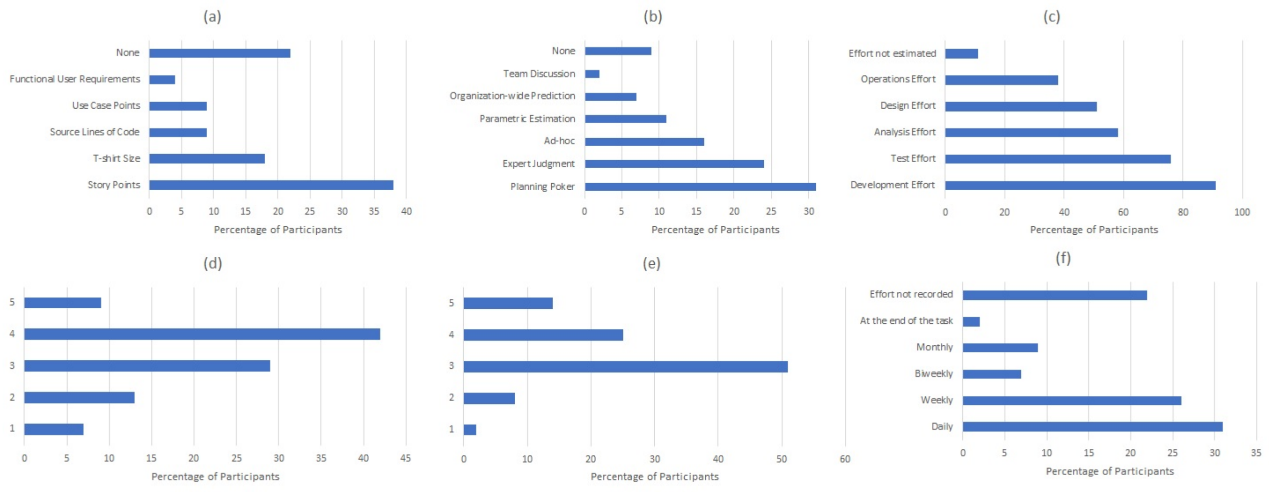

Figure 4. Participants responses regarding (a) software size measurement methods, (b) effort estimation techniques, (c) included tasks in effort estimation, (d) precision of the effort prediction, (e) changes in the efficiency of the effort estimation and (f) actual individual effort record frequency.

We can conclude that majority of the participated organizations perform size measurement and effort estimation while working with microservice-based projects. In general, they do not see any changes in the efficiency of the effort estimation process when utilizing microservice architectures. Also, the results show that the participants do not use functional size measurement (FSM) methods; most organizations apply subjective size measurement methods.

5. Discussion

In this paper, we presented the results of our survey on analysis, design, size measurement, and effort estimation of microservice-based architectures (The summary of the results in a tabular form can be found at https://bit.ly/3wzmzQZ). Our results show that there is no common approach utilized for these tasks among the participants. In general, the organizations utilize traditional software analysis and design notations and subjective size measurement and effort estimation methodologies for microservice-based projects.

Microservice-based architecture differs from traditional monolithic applications in many ways. The structural decompositions required by microservices are quite different from those of OOAD-based decomposition. The approach for the analysis and design approach of microservices should meet characteristics such as small bounded context, asynchronous communication, loose coupling, cohesion, isolation, autonomy, composability, single responsibility, scalability, and fault tolerance. Overall, OOAD, by its nature, may not have useful viewpoints to analyze and design a microservice-based solution that meets these essential characteristics. Thus, we believe there is a need for new modeling approaches for the analysis and design of microservices. For this purpose, we proposed an event-oriented analysis and design (EOAD) methodology [7] for microservices; which is based on extended event-driven process chain (eEPC) modeling notation. Our study shows the applicability of event-based modeling for the analysis and design of microservice-based projects.

The majority of the organizations apply subjective size measurement (such as story points) and effort estimation (such as planning poker) methods while working with microservices. However, these subjective methods have been criticized for their complexity, being specific to the dynamics of the team it is used in, and being difficult to use between teams. On the other hand, functional size measurement (FSM) methods provide objectivity to overcome these problems. These methods have been widely used in data-oriented architectures; however, microservice-based architectures are fundamentally different from the software that FSM methods are designed for. Since these methods are based on data-oriented measurement, they have data-oriented base measurement units. As a result, they are as strong to size the new generation projects. There is a need for a method to measure the size of microservice-based projects, using events as base measurement units which are fundamental concepts of the new generation architectures. In [8], we compared the effectiveness of two size measurement methods in a microservice-based project: COSMIC FSM [28] and Event Point [29]. The results of this study showed that measuring the size of microservices using Event Point enables establishing more precise effort estimation model.

The participants of our survey were from 9 countries from 4 continents. We previously presented the early results from Turkey on analysis and design of microservices [30]. The comparison between [30] and this study shows that the results from Turkey are in parallel with results from 9 countries. We also performed a continental comparison of the survey results (see the comparison table at `https://bit.ly/3wzmzQZ`). We categorized the continents into three: Western Europe, Eastern Europe, and America. We did not include Asia and Australia as the number of participants from these continents was not significant. Overall, we did not see any significant change in the results based on the comparison between continents. The organizations perform traditional OOAD techniques for analysis and design and subjective size measurement and effort estimation techniques for microservice-based architectures in general.

In general, we have seen that our results are in parallel with the literature. However, we have observed some differences between the obtained results and that of some previous studies. The observed differences are not about core concepts; they are mainly related to the technologies deployed. There may be two reasons behind these differences. The first reason may be related to the publication year of these previous studies. As technology evolves rapidly, organizations utilize different tools in time. The second reason may be that these previous studies do not mainly reveal the organizational choices. The choices between academy and industry may be different.

Lastly, our study is not without limitations. The number of participants in the survey may be considered a limitation. We have done our best to increase our sample size while making sure that a single organization will not dominate the survey. We have only asked at most two persons in single organizations to complete the survey. This is one of the factors that limit the number of participants, we have also paid attention to make sure participants were from a wide range considering their countries, experience, and organization size. As a result we end up a good representation but limited number of participants. Although it is difficult to generalize the results as best practices, we believe we provide a meaningful snapshot of the software industry that utilizes microservices.

6. Conclusion

The concept of microservice-based architecture is still quite new and it is not uniquely understood by the organizations who utilize the approach. In this study, we performed a survey to explore the organizational practices on software analysis, design, size measurement, and effort estimation when working with microservices to depict insight of the organizations who have experience with microservice-based projects. Our survey has depicted several surprising results as well as potential research studies to be performed.

'Object' is not the most proper if not totally wrong abstraction to decompose software into Microservices. However, our results show that organizations utilize common OOAD modeling notations such as class diagrams and sequence diagrams for the analysis and design of microservices. We do not believe this result can be generalized as a best practice, the issue might be related to a lack of specific notations and methodologies for microservice-based architectures or resistance to change. As Weinberg stated 'People choose familiar over the comfortable' [31]. That is, a chicken and egg problem we are facing.

There is also no specific approach for size measurement and effort estimation for microservice-based architectures. We assume it is a natural corollary of the lack of analysis and design methods as the size measurement is usually based on core concepts of these methods.

In conclusion, we can state that software developers lack the most fundamental tools in their work to develop microservice-based software systems. Our study revealed the lack of research studies that explores the common practices for the size measurement and effort estimation, analysis, and design of microservices. Thus, the open research areas for researchers are as follows: (1) Organizations use traditional size measurement and effort estimation techniques for microservice-based solutions. The effectiveness of these techniques can be compared to microservice-based projects. (2) Microservices move away from being data-driven and evolve into an event-oriented or message-driven structure. It can be assumed that size measurement would also be different from data-oriented architectures. Thus, there is a need to explore new software size measurement methods based on the units that better define microservice-based architectures. (3) The majority of organizations still utilize object-oriented analysis and design notations. Thus, there is a need to explore new analysis and design methods that result in a natural decomposition strategy for developing microservice-based solutions considering bounded context and asynchronous communication.

Acknowledgement

We would like to thank the support by The Scientific and Technological Research Council of Turkey (TUBITAK) ARDEB 1001 [Project number: 121E389] program.

References

[1] Sampaio AR, Kadiyala H, Hu B, Steinbacher J, Erwin T et al. Supporting Microservice Evolution. In: 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME); 2017. pp.539-543. doi:10.1109/ICSME.2017.63

[2] Thönes J. Microservices. IEEE Software. 2015;32 (1):116-116. doi:10.1109/MS.2015.11

[3] Ivanchikj A, Pautasso C, Schreier S. Visual modeling of RESTful conversations with RESTalk. Softw Syst Model. 2018;17 (3):1031-1051. doi:10.1007/s10270-016-0532-2

[4] Di Francesco P, Lago P, Malavolta I. Architecting with microservices: A systematic mapping study. Journal of Systems and Software. 2019;150:77-97. doi:10.1016/j.jss.2019.01.001

[5] Larrucea X, Santamaria I, Colomo-Palacios R, Ebert C. Microservices. IEEE Software. 2018;35 (3):96-100. doi:10.1109/MS.2018.2141030

[6] Dragoni N, Giallorenzo S, Lafuente AL, Mazzara M, Montesi F et al. Microservices: Yesterday, Today, and Tomorrow. In: Mazzara M, Meyer B, eds. Present and Ulterior Software Engineering. Springer International Publishing; 2017;195-216. doi:10.1007/978-3-319-67425-4_12

[7] Unlu H, Tenekeci S, Yıldız A, Demirors O. Event Oriented vs Object Oriented Analysis for Microservice Architecture: An Exploratory Case Study. In: 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2021); 2021;244-251. doi: 10.1109/SEAA53835.2021.00038

[8] Unlu H, Hacaloglu T, Leblebici O, Demirors O. Effort Prediction for Microservices: A Case Study. In: 15. Turkish National Software Engineering Symposium (UYMS 2021); 2021;1-6. doi: 10.1109/UYMS54260.2021.9659766

[9] Bandeira A, Medeiros CA, Paixao M, Maia PH. We Need to Talk About Microservices: an Analysis from the Discussions on StackOverflow. In: 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR);2019. pp.255-259. doi:10.1109/MSR.2019.00051

[10] Soldani J, Tamburri DA, Van Den Heuvel W-J. The pains and gains of microservices: A Systematic grey literature review. Journal of Systems and Software. 2018;146:215-232. doi:10.1016/j.jss.2018.09.082

[11] Bogner J, Fritzsch J, Wagner S, Zimmermann A. Microservices in Industry: Insights into Technologies, Characteristics, and Software Quality. In: 2019 IEEE International Conference on Software Architecture Companion (ICSA-C);2019. pp.187-195. doi:10.1109/ICSA-C.2019.00041

[12] Ghofrani J, Lübke D. Challenges of Microservices Architecture: A Survey on the State of the Practice. In: ZEUS; 2018. pp.1-8.

[13] Zhou X, Peng X, Xie T, Sun J, Ji C et al. Fault Analysis and Debugging of Microservice Systems: Industrial Survey, Benchmark System, and Empirical Study. IEEE Transactions on Software Engineering. 2021;47 (2):243-260. doi:10.1109/TSE.2018.2887384

[14] Taibi D, Lenarduzzi V, Pahl C. Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation. IEEE Cloud Computing. 2017;4 (5):22-32. doi:10.1109/MCC.2017.4250931

[15] Carvalho L, Garcia A, Assunção W, Mello R, Julia de Lima M. Analysis of the Criteria Adopted in Industry to Extract Microservices. In: 2019 IEEE/ACM Joint 7th International Workshop on Conducting Empirical Studies in Industry (CESI) and 6th International Workshop on Software Engineering Research and Industrial Practice (SER IP);2019. pp.22-29. doi:10.1109/CESSER-IP.2019.00012

[16] Di Francesco P, Lago P, Malavolta I. Migrating Towards Microservice Architectures: An Industrial Survey. In: 2018 IEEE International Conference on Software Architecture (ICSA);2018. pp.29-2909. doi:10.1109/ICSA.2018.00012

[17] Balalaie A, Heydarnoori A, Jamshidi P, Tamburri DA, Lynn T. Microservices migration patterns. Software: Practice and Experience. 2018;48 (11):2019-2042. doi:https://doi.org/10.1002/spe.2608

[18] Viggiato M, Terra R, Rocha H, Valente MT, Figueiredo E. Microservices in Practice: A Survey Study. In VEM 2018-6th Workshop on Software Visualization, Evolution and Maintenance, Sep 2018, Sao Carlos, Brazil.

[19] Vural, H, Koyuncu, M, Guney, S. A systematic literature review on microservices. In International Conference on Computational Science and Its Applications (pp. 203-217). Springer, Cham.

[20] Shull F, Singer J, Sjøberg DI. Guide to Advanced Empirical Software Engineering. Springer, 2007.

[21] Linaker J, Sulaman SM, Höst M, de Mello RM. Guidelines for Conducting Surveys in Software Engineering v. 1.1. vol; 2015.

[22] O'Connor RV, Elger P, Clarke PM. Continuous software engineering—A microservices architecture perspective. Journal of Software: Evolution and Process. 2017;29 (11):e1866. doi:10.1002/smr.1866

[23] Famelis M, Chechik M. Managing design-time uncertainty. Softw Syst Model. 2019;18 (2):12249-1284. doi:10.1007/s10270-017-0594-9

[24] Malavolta I, Lago P, Muccini H, Pelliccione P, Tang A. What Industry Needs from Architectural Languages: A Survey. IEEE Transactions on Software Engineering. 2013;39 (6):869-891. doi:10.1109/TSE.2012.74

[25] Alshuqayran N, Ali N, Evans R. A Systematic Mapping Study in Microservice Architecture. In: 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA);2016. pp.44-51. doi:10.1109/SOCA.2016.15

[26] Salmanoğlu M, Öztürk K, Bağrıyanık S, Ungan E, Demirörs O. Benefits and challenges of measuring software size: early results in a large organization. In: 25th International Workshop on Software Measurement and 10th International Conference on Software Process and Product Measurement, IWSM-Mensura;2015.

[27] Demirors O, Gencel C. Conceptual Association of Functional Size Measurement Methods. IEEE Software. 2009;26 (3):71-78. doi:10.1109/MS.2009.60

[28] COSMIC Measurement Manual for ISO 19761 - v5.0: Common Software Measurement International Consortium (COSMIC),2021.

[29] Hacaloglu T. Event Points: A Software Size Measurement Model. PhD, Middle East Technical University, Ankara, Turkey, 2021.

[30] Bilgin B, Unlu H, Demirörs O. Analysis and Design of Microservices: Results from Turkey. In: 2020 Turkish National Software Engineering Symposium (UYMS) pp. 1-6 doi: 10.1109/UYMS50627.2020.9247022

[31] Weinberg GM. Quality Software Management (Vol. 1) Systems Thinking. Dorset House Publishing Co., Inc., 1992.