MICROM: A SIZE MEASUREMENT METHOD FOR MICROSERVICE-BASED ARCHITECTURES

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements of the Degree of

DOCTOR OF PHILOSOPHY

in Computer Engineering

by
Hüseyin ÜNLÜ

December 2024
İZMİR

We approve the thesis of **Hüseyin ÜNLÜ**

Examining Committee Members:

Prof. Dr. Onur DEMİRÖRS
Department of Computer Engineering, İzmir Institute of Technology

Prof. Dr. Tolga AYAV
Department of Computer Engineering, İzmir Institute of Technology

Prof. Dr. Oğuz DİKENELLİ
Department of Computer Engineering, Ege University

Assoc. Prof. Dr. Rıza Cenk Erdur
Department of Computer Engineering, Ege University

Assist. Prof. Dr. Emrah İNAN
Department of Computer Engineering, İzmir Institute of Technology

12 December 2024

Prof. Dr. Onur DEMİRÖRS
Supervisor Department of Computer
Engineering, İzmir Institute of Technology

Prof. Dr. Onur DEMİRÖRS
Head of Department of Computer
Engineering, İzmir Institute of Technology

Prof. Dr. Mehtap EANES
Dean of the Graduate School

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Dr. Onur Demirörs, for his invaluable guidance, unwavering support, and encouragement throughout my research. His expertise and mentorship have been instrumental in shaping my academic life.

I am sincerely thankful to my thesis monitoring committee members, Prof. Dr. Tolga Ayav and Prof. Dr. Oğuz Dikenelli, for their insightful feedback and constructive suggestions during my monitoring meetings. I am also deeply grateful to jury members, Assoc. Prof. Dr. Rıza Cenk Erdur and Assist. Prof. Dr. Emrah İnan, for their thoughtful review and valuable comments.

This thesis was supported by TÜBİTAK ARDEB 1001 [Project Number: 121E389] program, and I extend my gratitude for their financial support.

I express my special thanks to my colleague Tuna Hacaloğlu, whose opinions have made significant contributions to my work. I am also deeply thankful to my friends, who not only believed in me but also shared countless moments of laughter and joy throughout this journey. A special mention goes to the Gorillas: Samet Tenekeci, Eren Uzyıldırım, and Altuğ Yiğit.

My sincere thanks go to my mother Emine Betül Aksoy, my father Ahmet Ünlü, and my brother Ata Ünlü for their love, encouragement and support throughout my life. I also extend my heartfelt thanks to Ulviye Sareç, Şerif Sareç, Ünzile Yanar, Savaş Yanar, İrem Yanar, and Kaan Yanar for their kindness, support, and belief in me throughout this journey.

I owe my most special thanks to my wife Merve. I am extremely grateful for motivating and supporting me and not letting me give up. Thank you for loving me, believing in me and always supporting me.

ABSTRACT

MICROM: A SIZE MEASUREMENT METHOD FOR MICROSERVICE-BASED ARCHITECTURES

The paradigm shift in architectures has led the new generation of software projects, such as microservice-based software architecture (MSSA), to move away from being data-driven and to evolve into a behavior-oriented structure. The usage of a single database is replaced by the structures in which each microservice is developed independently and has its own database. While traditional monolithic architectures rely on functions for data transfer, MSSA uses structures like event queues for communication. As a primary input for effort and cost estimation, Functional Size Measurement (FSM) methods have been used widely for objective size measurement of monolith software architectures. However, these methods may not respond to the size measurement of today's applications, such as MSSA, as they are based on data movements. In this thesis, we proposed a size measurement method called MicroM, developed explicitly for MSSAs, which uses events, the base components of MSSAs, for size measurement. MicroM considers the characteristics of MSSAs and addresses the shortcomings of existing FSM methods. The proposed method uniquely categorizes events into functional, architectural, and algorithmic levels. We evaluated the success of the proposed method by comparing it with the COSMIC FSM method in effort estimation using real-world data across three different case studies. MicroM achieved an improvement of up to 32% in Mean Magnitude of Relative Error (MMRE) in the effort estimation models compared to the COSMIC FSM method.

ÖZET

MICROM: MİKROSERVİS TABANLI MİMARİLER İÇİN BÜYÜKLÜK ÖLÇÜM YÖNTEMİ

Yazılım mimarisindeki paradigma değişimi, mikroservis tabanlı yazılım mimarisi (MSYM) gibi yeni nesil yazılım projelerinin veri odaklı olmaktan uzaklaşmasına ve davranış odaklı bir yapıya evrilmesine neden olmuştur. Tek bir veri tabanı kullanımı, her mikroservisin bağımsız olarak geliştirildiği ve kendi veri tabanına sahip olduğu yapılarla değiştirilmiştir. Geleneksel monolitik mimariler veri transferi için fonksiyonlara dayanırken, MSYM iletişim için olay kuyrukları gibi yapılar kullanmaktadır. Efor ve maliyet kestirimi için birincil girdi olarak İşlevsel Büyüklük Ölçüm (İBÖ) yöntemleri, monolitik yazılım mimarilerinin nesnel boyut ölçümü için yaygın olarak kullanılmıştır. Ancak, bu yöntemler veri hareketlerine dayandıkları için günümüz uygulamalarının, örneğin MSYM'nin büyüklük ölçümüne yanıt vermeyebilir. Bu tezde, MSYM'ler için özel olarak geliştirilmiş, MSYM'nin yapı taşları olan olayları büyüklük ölçümü için kullanan MicroM adlı bir büyüklük ölçüm yöntemi önerilmiştir. MicroM, MSYM'nin özelliklerini dikkate alır ve mevcut İBÖ yöntemlerinin eksikliklerini giderir. Önerilen yöntem, olayları işlevsel, mimari ve algoritmik düzeylere özgün bir şekilde kategorize eder. Önerilen yöntemin başarısını, gerçek organizasyon verileri kullanarak üç farklı vaka çalışmasında COSMIC İBÖ yöntemiyle karşılaştırarak değerlendirilmiştir. MicroM yöntemi ile oluşturulan efor kestirim modellerinde, COSMIC İBÖ yöntemine kıyasla Ortalama Göreli Hata (MMRE) oranında %32'ye varan iyileşme sağlanmıştır.

to my lovely wife, Merve

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

DDD	Domain Driven Design
DM	Data Movement
DSR	Design Science Research
eEPC	extended Event-Driven Process Chain
EPC	Event-Driven Process Chain
ER	Entity Relationship
FP	Functional Process
FPA	Function Point Analysis
FSM	Functional Size Measurement
FUR	Functional User Requirement
HSEC	Human Subjects Ethics Committee
MdMRE	Median Magnitude of Relative Error
MIS	Management Information System
MMRE	Mean Magnitude of Relative Error
MRE	Magnitude of Relative Error
MS	Microservice
MSSA	Microservice-based Software Architecture
OOI	Object of Interest
PBI	Product Backlog Item
PRED	Percentage Relative Error Deviation
REST	REpresentational State Transfer
SDLC	Software Development Life Cycle
SOA	Service Oriented Architecture
SOAP	Simple Objects Access Protocol
SP	Story Point
UCP	Use Case Point
UML	Unified Modeling Language

CHAPTER 1

INTRODUCTION

This thesis aims to develop an objective size measurement method tailored explicitly for microservice-based software architectures (MSSA). This chapter provides an overview of the key elements guiding this thesis. It begins with a detailed problem statement, highlighting the challenges that motivate this research. Following that, the chapter outlines the objectives and contributions of the study, focusing on the development of an objective size measurement method for microservice-based architectures. Next, the summary of the research strategy is presented, explaining the approach and methodology employed to achieve the study's goals. Finally, the chapter concludes with the thesis outline, offering a roadmap of the subsequent chapters and their content.

1.1. Problem Statement

The software industry is currently undergoing a significant paradigm shift. Over the past decade, MSSA has emerged as a favored design paradigm for an increasing number of organizations. First introduced in 2011, this paradigm evolved in response to the new demands from modern software systems.[1] Since 2015, MSSA's popularity has surged, with major service providers like Amazon, LinkedIn, Netflix, SoundCloud, Uber, and Verizon embracing this architecture.[2]

MSSA promotes building systems using a collection of small, independent services. Each service manages its own data and is designed to be isolated, scalable, and resistant to failure. These services work together to create a unified system that is much more flexible and resilient than the traditional monolith systems commonly developed during the past decade.[3] They can be developed, deployed, tested, and scaled independently.[2,4,5]

MSSA can be associated with Service-Oriented Architecture (SOA). The relationship between SOA and MSSA is rooted in their shared foundational goals, such as decoupling, isolation, composition, integration, and the use of discrete and autonomous services. In this sense, MSSA can be seen as an evolution of SOA, as both aim to achieve similar objectives. However, MSSA differs from SOA in three key aspects: size, bounded context, and autonomy.[6] In MSSA, each microservice focuses on delivering a single business capability. Second, it consolidates all related functionalities within that capability. Third, because each microservice has a specific, defined responsibility, they operate as independent services. Changes in one microservice should not impact others. MSSA is often linked with event-driven asynchronous communication and ledger-style data persistence, which enhances scalability, reliability, and performance in real-world applications.[3,7,8] The use of events in asynchronous communication between services enables faster decision-making, lowers costs, and improves scalability in microservices.[9]

MSSA naturally complements the Agile development methodology.[10] In 2001, the Agile Manifesto was introduced as a response to the challenges posed by traditional software development approaches.[11] Since then, Agile has become widely adopted by organizations looking to streamline and modernize their software development processes.[12] One of the fundamental principles of the Agile Manifesto is that delivering working software frequently can be achieved through independent microservices, each focused on a single business capability. Additionally, agile teams are likely to find it easier to understand domain concepts within a limited context, given that these services are related to one another.[10] However, organizations must balance decentralization and autonomy with the effective management and integration of outputs from all teams to deliver software on time and within budget.

In a software project, meeting customer expectations on time and within the planned budget is achieved through effective project management, especially with precise effort estimation.[13] As the primary input for effort estimation, size measurement methods can be divided into two main categories: formal methods and expert opinion-based methods.[14]

In today's Agile world, subjective (expert opinion-based) methods like Story Point (SP) and Use Case Point (UCP) are more commonly used than objective (formal) methods such as Functional Size Measurement (FSM) methods.[15] Supporters of subjective size measurement often point out the simplicity and the flexibility they provide to estimators.[16] The common use of SP among agile practitioners is due to their adaptability, allowing teams to estimate the size of functional requirements and a variety of tasks, including social and technical efforts.[17] Practitioners also use SP to indicate different concepts such as size, effort, time, and complexity.[16] Their versatility means that using SP doesn't require special training or extra costs. Additionally, agile teams value self-organization and want control over decision-making, which aligns well with the subjective nature of SP.[16,18] On the other hand, they are criticized for issues such as misinterpretation of size, complexity, subjectivity, and lack of transferability between teams.[16,19–22]

FSM-based methods provide an objective process to mitigate these drawbacks. FSM not only aids in estimating effort and cost but also serves other purposes, such as controlling and monitoring project scope and risks, evaluating process performance, and establishing standardized measures at the organizational level.[23] FSM has a long history that spans nearly five decades. Over this time, the concept has evolved into widely accepted methods.[24,25] Several FSM methods are recognized as ISO standards: IFPUG CPM[26], FiSMA[27], NESMA[28], MK II[29], and COSMIC[30]. In the literature, many studies[18,31–33] show successful use of FSM-based objective size measurement methods in agile settings.

FSM methods have been widely used for objective size measurement of monolithic software architectures. However, these methods may not adequately address the size measurement of modern applications, such as MSSA. MSSA moved away from being data-driven and evolved into a behavior-oriented structure. The usage of a single database is replaced by the structures in which each microservice is developed independently and has its own database. With the paradigm shift, today's systems differ from the software for which FSM methods are designed. For instance, COSMIC, a second-generation FSM method, illustrates this mismatch as follows.

COSMIC FSM aims to count data movements associated with a specific object of interest by combining functionality with that object, defining project size as the total of these data movements. It recommends using techniques like Entity Relationship (ER) diagrams, Unified Modeling Language (UML) class diagrams, or relational data analysis to identify these objects.[34] In contrast, MSSA typically utilizes non-relational NoSQL databases due to their distributed nature.[35] While traditional monolithic architectures rely on functions for data transfer (such as method parameters), MSSA uses structures like event queues for communication. MSSA requires a more behavioral structure over service calls/requests (such as REST API) instead of a structural form such as data flow to integrate the services.[36] Consequently, the relational data-based approach is rarely applied in MSSA, making it challenging to achieve efficient outcomes, such as accurate effort or cost estimations, when this approach underpins software size measurement.

The literature search showed a lack of studies on size measurement in MSSA. We also found that no de facto method is used in the industry for microservice-based (MS-based) projects.[37,38] However, the increasing prevalence of MSSA in the software industry has led to the need for a new base component in software size measurement. The concept of data movement in FSM methods found in the literature is no longer a fundamental element of MSSA. Consequently, organizations face unique challenges in size measurement and effort estimation for these projects. All these findings led to the need to develop a size measurement method tailored explicitly for MSSA. The following section summarizes the objectives and contributions of the proposed size measurement method: MicroM.

1.2. Objectives and Contributions

The shift from database and transaction-oriented structures to behavioral and event-oriented structures in the software world, driven by the ongoing paradigm change, has highlighted the inadequacy of existing FSM methods in addressing this change and the lack of a size measurement method specifically developed for this paradigm. This paradigm shift has led to the need to develop a unique size measurement method for MSSAs.

A size measurement method developed specifically for an architecture should define size in terms of the base components of that architecture. To illustrate, the COSMIC method, successfully used in database and transaction-oriented monolithic structures, expresses software size through data movement units.[25] In an MSSA, the base components can be expressed as events.[3] Therefore, we aimed to develop an event-based size measurement method for MSSAs.

Categorizing events into different abstraction levels of the software representations while considering the characteristics of the architecture is essential for creating a method applicable at every stage of the SDLC. For example, during the requirements phase, the communication between microservices remains undefined. However, these details become clearer as we advance through the SDLC phases.

Therefore, we defined the event levels in the developed method by considering the details identified during the SDLC phases in MS-based projects.

This thesis contributes to the literature by introducing an event-based size measurement method called MicroM that considers the characteristics of MSSAs and addresses the shortcomings of utilizing existing FSM methods on event-based systems. MicroM, specifically developed for MSSAs, uniquely categorizes the concept of events into abstraction levels of the software representations while considering the characteristics of microservice architecture. Accordingly, this thesis provides the following contributions:

- Revealing the size measurement and effort estimation methods used in MS-based projects within the software industry, as well as the challenges encountered during this process
- Evaluating the relevance of using events as a size measure in MS-based projects
- Developing a new size measurement method that categorizes the concept of events into abstraction levels of the software representation, considering the characteristics of MSSAs, and can be applied at every stage of the SDLC
- Evaluating the success and relevance of the developed method in real-world cases

MicroM aims to reduce organizations' challenges in the software industry when measuring size and estimating effort in their MS-based projects. Organizations can measure the size of their MS-based projects based on the events that are the base components of the architecture, define this size at different abstraction levels of the software representation, and estimate effort based on various size levels or the total size.

1.3. Research Strategy

In this thesis, we aimed to develop a method to measure the size of MS-based projects in terms of base components: events. We formulated the following research questions for this aim:

- *RQ1. What are the common practices and challenges encountered in software size measurement and effort estimation in MS-based projects?*
- *RQ2. Is "event" relevant as a size measure in MSSAs?*

- *RQ3. What types of events can be used as base components at different levels of software representation?*
- *RQ4. How successful is the proposed MicroM size measurement method for effort estimation in MS-based projects?*

We used Design Science Research (DSR) as our research methodology to answer the research questions we identified. We followed the guidelines for DSR proposed by Hevner et al.[39] and the DSR process model suggested by Offermann et al.[40]. In the Problem Identification phase, we conducted a literature review, survey, and interview to answer RQ1 (see Chapter 2, Sections 4.1, and 4.2). For RQ2, we performed an exploratory case study on event-based size measurement in an MS-based project (see Section 4.3). In the Solution Design phase, we developed our proposed MicroM method iteratively and incrementally to address RQ3 (see Sections 4.4, 4.5, and Chapter 5). Finally, in the Evaluation phase, we assessed the success of the MicroM method through three separate case studies to answer RQ4 (see Chapter 6).

1.4. Thesis Structure

This thesis is structured as follows: Chapter 2 summarizes the related work. Chapter 3 describes the research methodology. Chapter 4 presents the development process of the size measurement method. Section 5 describes the proposed size measurement method. Section 6 presents the evaluation of the proposed method. Lastly, Section 7 concludes the thesis by stating further research suggestions.

CHAPTER 2

LITERATURE REVIEW

This chapter first provides essential background information and summarizes the related work in the following sections.

2.1. Background

In this section, we provide the essential background on (1) MSSA, (2) size measurement methods involved in the thesis (COSMIC FSM and Event Point), and (3) the extended Event-Driven Process Chain (eEPC) modeling notation proposed by MicroM method for identifying events.

2.1.1. Microservice-based Software Architectures (MSSA)

The foundations of MSSA began to emerge in the early 2000s, influenced by SOA. In 2005, Peter Rodgers introduced the term “Micro Web Services” and pioneered the adoption of REpresentational State Transfer (RESTful) services during a period when Simple Objects Access Protocol (SOAP) was predominant.[41] However, due to technological limitations, this concept remained largely theoretical for a period. In March 2014, Martin Fowler's article titled “Microservices”, published on his website, significantly impacted the understanding and adoption of microservices within the software development community.[42] Following this, Netflix adopted a microservice architecture to enhance scalability and resilience. Additionally, cloud computing services like Amazon Web Services (AWS) have become indispensable facilitators for microservices. With the launch of Docker containers, which provide a lightweight and consistent environment for smooth transitions between development and production, and Kubernetes, a container orchestration platform, the deployment, scaling, and management of containerized applications became simpler, promoting microservices.[5,6,43] Major technology organizations, including Amazon, Google, and Microsoft, embraced microservices to enhance agility.[2] The emergence of microservice design patterns, serverless computing, and event-driven tools have made MSSA a popular architecture in today's software world.[1]

In MSSA, communication between microservices can occur in a request-driven or event-driven way.[3,7,8,36,44,45] Request-driven MSSA relies on a synchronous communication model. When one service needs data or functionality from another, it sends a direct request and waits for a response before continuing its process. On the other hand, event-driven MSSA uses an asynchronous communication model uses events to trigger and communicate between decoupled microservices. In this approach, services publish events when specific actions occur, and other services can subscribe to these events. This allows services to respond to changes in real time without direct dependencies.

Although event-driven architecture enables real-time communication, a large amount of data handling, and data updates in real-time, request-driven architecture may be more suitable for specific scenarios, such as authentication, where a request-response model is better suited for managing security. Choosing between request-driven and event-driven microservices depends on the response time requirements, scalability needs, complexity tolerance, coupling requirements, and development speed.

2.1.2. Size Measurement Methods

We used the COSMIC FSM for comparison purposes and the Event Point to evaluate the relevance of an event-based measurement method in MSSA during the development of the proposed MicroM method.

2.1.2.1. COSMIC FSM

In 1979, Albrecht introduced Function Points (FP) and Function Point Analysis (FPA) with the goal of measuring software size based on the "functionality" it offers to users.[47] Following this, various functional size measurement methods were developed using the same principle. Several FSM methods are recognized as ISO standards: IFPUG CPM[26], FiSMA[27], NESMA[28], MK II[29], and COSMIC[30]. While all these FSM methods use Functional User Requirement (FUR) to calculate size, they differ in their measurement approach and units.[25]

As a second-generation FSM method, COSMIC FSM is based on data movements.[34] For this purpose, the method suggests identifying FURs and the functional processes (FP) related to these requirements and the data movements (DM) within these processes. These data movements are identified as the flow of data groups that define a single Object of Interest (OOI). In other words, for a movement to be classified as a DM, it must be associated with a single OOI. DM can appear in four different forms: Entry (E), Exit (X), Read (R), and Write (W). Figure 2.1 illustrates the DM types and their relationship with a functional process.

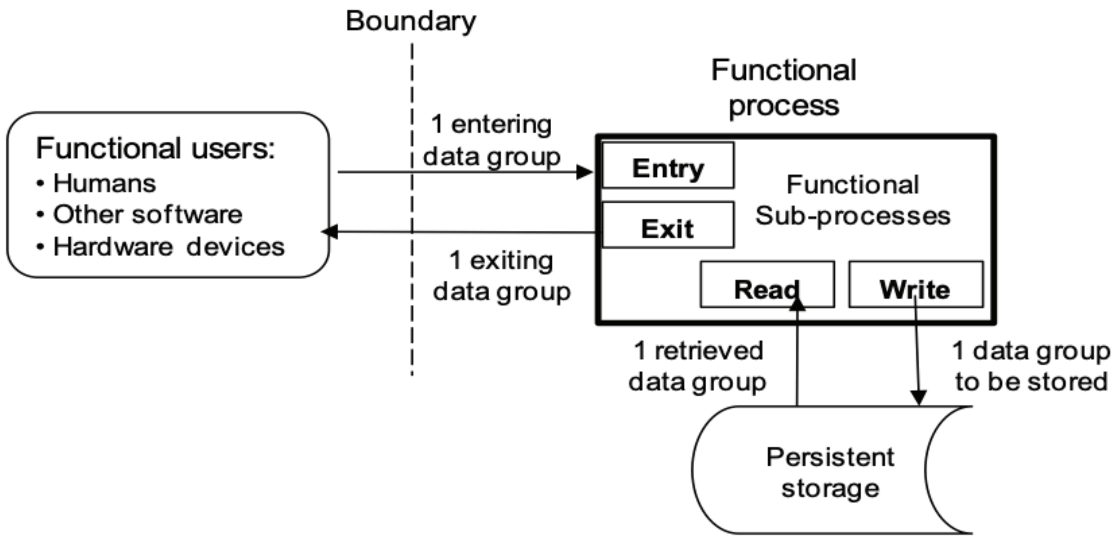

Figure 2.1. The four types of data movement[34].

COSMIC FSM defines the unit of measurement as the COSMIC Function Point (CFP), which is assigned to each identified DM in a functional process. The size of a functional process is determined by summing all the identified DMs within that process. To calculate the size of each software component within a layer, the size of its functional processes is aggregated based on the identified FUR for the software.

COSMIC FSM provides various measurement guidelines with varying levels of detail for measuring software size in different domains, such as business application software[48], real-time software[49], and service-oriented software[50]. The method is applicable in data-rich (such as management information system (MIS) applications), control-rich (such as elevators, washing machines, and cars), and hybrid (such as real-time reservation systems) software domains. On the other hand, it is not applicable in algorithm-rich (such as expert systems, simulation software, and weather forecasting systems) and large file processing-rich (such as games and musical instruments) due to its data movement-based measurement strategy. The method can be used at every stage of the SDLC, but as the measurement artifacts mature, the accuracy of the measurements improves.

2.1.2.2. Event Point

Event Point[51] provides an alternative approach to software sizing compared to traditional data-centric methods like COSMIC FSM. The method is based on events triggered by functional user interactions with the system. The method introduces the concept of a computation event. An event is generated as a result of computation activities in the system, such as display, recording, calculation, retrieval, decision, and communication. Event Point categorizes computation events as “System Boundary Event,” “Display Event,” “Calculation/Processing Event,” “Record Event,” “Decision Event,” “Retrieval Event,” and “Communication Event.”

The method suggests drawing eEPC diagrams to identify events. Each computation event is assigned one Event Point, similar to how COSMIC FSM assigns equal weight to data movements. A system is expected to consist of a sequence of events, so the size of a functional user requirement is the total number of computation events it includes. Thus, the software's total size is the sum of the Event Points across all functional requirements.

Event Point method's “System Boundary,” “Retrieval,” “Record,” and “Display” events can be mapped to COSMIC FSM's “Entry,” “Read,” “Write,” and “Exit” data movements, respectively. The distinction between DMs and events is that while COSMIC FSM associates each DM with a data group linked to a specific object of interest, event-based definitions do not follow a data-centric approach, allowing the measurer to disregard the software's data structure and model. The method can be used at every stage of the SDLC, similar to COSMIC FSM.

The Event Point method is the first to propose events as a software size measure. It has proven more relevant and provides more accurate effort estimation than the traditional COSMIC FSM method in contemporary software projects.[52] However, the method has been developed generally for modern software and does not consider the characteristics of MSSA.

2.1.3. Extended Event-Driven Process Chain (eEPC)

The extended Event-Driven Process Chain (eEPC) is based on the Event-Driven Process Chain (EPC) modeling notation developed by the University of Saarland in Germany in 1992.[53] The essential elements of EPC are events, functions, and control flows. A function is defined as "an activity (task, process step) which needs to be executed," while an event is "the pre-/post-condition of a function". Due to its readability, user-friendliness, and ease of modeling, the EPC modeling notation has been used by business users for more than 30 years.[54,55] Since the development of EPC, various versions have been created, and eEPC is one of these 14 variations.[53,56] eEPC extends the EPC, which includes functions, events, and connectors, by adding elements such as "organizational units," "information objects," and "IT systems". Figure 2.2 illustrates the major element connections in eEPC modeling notation.

In the literature, several studies on EPC emphasize its role in software engineering. Lübke[58] proposed a procedure to transform use cases into EPC models, addressing the lack of control flow in use case diagrams. Gross and Doerr[59] investigated the effectiveness of EPC and Activity Diagrams in requirements engineering, showing that both methods can support this process. Dragicevic et al.[55] developed a method for Agile software development, using eEPC for elicitation, documentation, and validation of user requirements. Amjad et al.[60] argued that EPC is suitable for modeling and verifying simple business requirements, though it struggles with complex event processing, leading to limitations in modeling more complex systems.

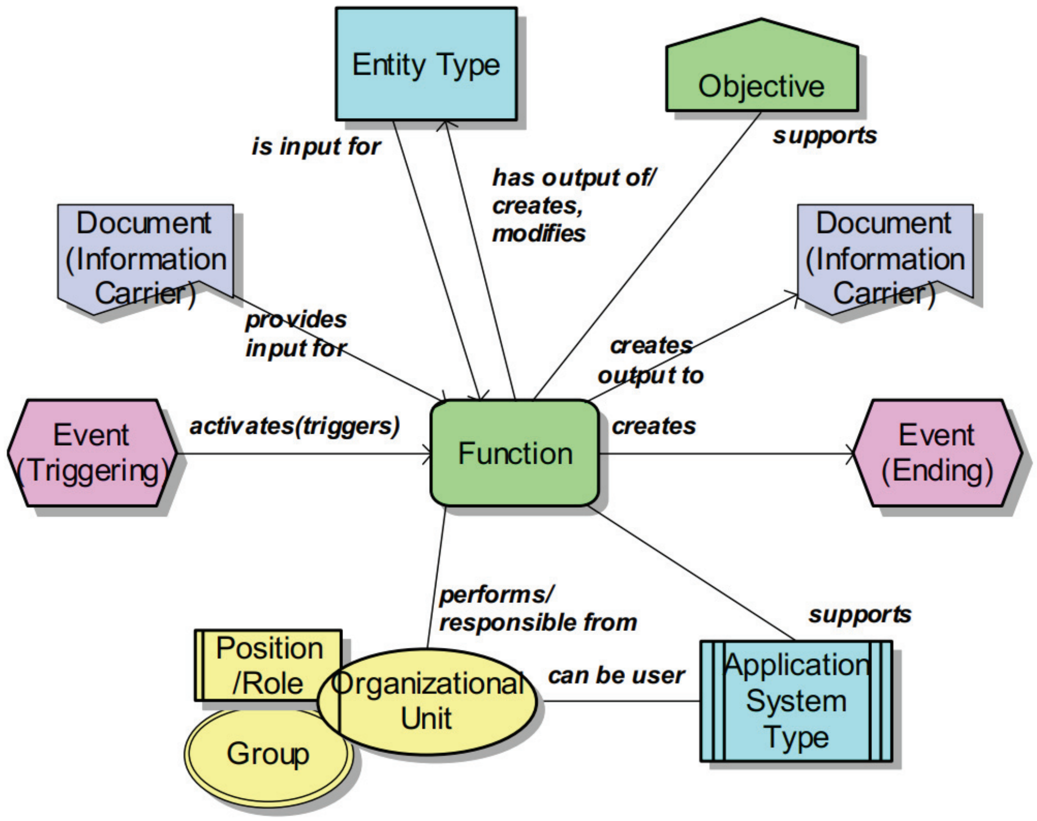

Figure 2.2. Major element connections in eEPC modeling notation[57].

The MicroM method developed in this thesis proposes using eEPC notation to identify events, highlighting its compatibility with the event-driven approach of MSSA. Existing literature also supports the recommendation of eEPC for analysis and design within microservice-based projects. We previously compared the effectiveness of object-oriented and event-oriented approaches for analyzing and designing microservice-based systems.[7] We modeled project requirements for event-oriented analysis and design using eEPC. We observed that the event-oriented approaches offer significant advantages over the object-oriented approaches, particularly in light of the characteristics of MSSA.

2.2. Related Work

In the literature, a limited number of studies attempted to measure the size of MS-based systems. Asik and Selcuk[61] proposed a size measurement framework for MSSA that counts resources and clients responsible for interactions between microservices and external services. Vural, Koyuncu, and Misra[62] measured the size of a monolith using the COSMIC FSM method in a case study and then measured the size of the microservices created by splitting this monolith using Domain Driven Design (DDD). Taibi and Systa[63] proposed a measurement framework for microservices based on coupling, number of classes, number of duplicated classes, and frequency of external calls, associating the number of classes and duplicated classes with size.

The literature review showed that the proposed size measurement methods require design-time decision elements such as coupling, number of classes, resource, and client count. However, a software size measurement method needs to be applicable at every stage of the SDLC for effective project management. Project size should be measurable starting from the early requirements phase. Furthermore, none of these studies evaluated the success of their proposed method in activities such as effort estimation, cost estimation, or project planning, where software size is used as the primary input. Consequently, despite MSSA's presence in the software industry for over a decade, the literature lacks an established size measurement method specifically for MSSA, similar to the recognized methods such as COSMIC[30] or IFPUG[26] used for traditional monolithic architectures.

Existing FSM methods can be mapped to MSSA. For instance, IFPUG published a white paper that describes the application of Function Point Analysis (FPA) to microservices-based systems.[64] Similarly, MS-based systems can also be measured using the communication rules between services discussed in the COSMIC SOA Measurement Guideline, which suggests counting requests as exits and responses as entries.[50] FSM methods are architecture-independent; however, in the current paradigm shift, the data-oriented systems for which these methods were developed have evolved into behavior-oriented systems, with service calls/requests replacing data flow and event queues taking the place of functions for data transfer. In MSSA, events, rather than data, form the base components.

The event, which forms the foundation of the proposed MicroM method, is articulated in philosophy as the "exemplification of objects or properties of objects within the context of the time". Event theory has been examined in different ways by philosophers.[65] Jaegwon Kim[66] suggested that events are essentially examples of properties. In contrast, Donald Davidson[67] identified events based on their causes and effects but later changed his view to define events by their location in spacetime. David Lewis[68–71] described an event as a property of a specific area in space and time. Choosing a theory of events is not something decided independently; it is influenced by one's other philosophical beliefs and interests.

The concept of events has been applied in software engineering literature across different types and levels of abstraction such as atomic[60], business[72], complex[60], composite[73,74], computation[51], domain[75,76], external[72,77], hardware[78], primitive[72–74], state describing[77], software[77], system[72,79], triggering[34,60], trivial[80], and user[72,77,79] events. However, no study in the literature categorizes the concept of events into different levels of abstraction, considering the characteristics of MSSA.

2.3. Summary of the Chapter

This section provided a concise overview of the essential background on (1) MSSA, (2) the size measurement methods discussed in the thesis (COSMIC FSM and Event Point), and (3) the extended Event-Driven Process Chain (eEPC) along with related research. The literature review indicates that only a few studies have attempted to measure the size of MS-based systems, and no method has been explicitly developed for this purpose. The subsequent section outlines the research methodology adopted in this thesis.

CHAPTER 3

RESEARCH METHODOLOGY

Software Engineering, as an interdisciplinary field, integrates both technical and social aspects, making empirical methods crucial due to the human-intensive nature of software development.[81] Research in software engineering can be categorized into two main categories: (1) Empirical Research and (2) Design Science Research (DSR).[82] Empirical research focuses on answering exploratory, base rate, and relationship questions, while DSR addresses design-oriented questions, such as determining the most effective method for achieving a specific objective through an artifact. While both methodologies can be combined in a research study, DSR is considered superior as it employs empirical research to evaluate solutions. DSR's advantage lies in its suitability for cases where problem-solving requires creating a solution through an iterative, flexible, and exploratory process.[82]

This thesis adopts DSR to develop a novel software size measurement model for MSSAs. The human-made nature of software engineering artifacts aligns well with DSR, which focuses on understanding and improving human-created designs.[83] Hevner et al.[39] first introduced DSR for information systems, and Wieringa[84] later expanded its use in software engineering. DSR seeks to create innovative artifacts that enhance human and organizational capabilities. The goal of this thesis involves designing a new size measurement model for MSSAs; therefore, DSR's proactive and technology-driven approach fits this research well. Following the guidelines by Hevner et al.[39] and the research process outlined by Offermann et al.[40], we structured our research methodology.

Hevner et al.[39] provide useful insights into the nature of the DSR process in their proposed guidelines. However, the suggested guideline does not specify the steps a research study adopting DSR should follow. Offermann et al.[40] highlighted the lack of a clear approach for integrating a research methodology within DSR and proposed a detailed research process that formally combines qualitative and quantitative methods. The process proposed by Offermann et al.[40] consists of three iterative phases: (1) Problem Identification, (2) Solution Design, and (3) Evaluation (see Figure 3.1).

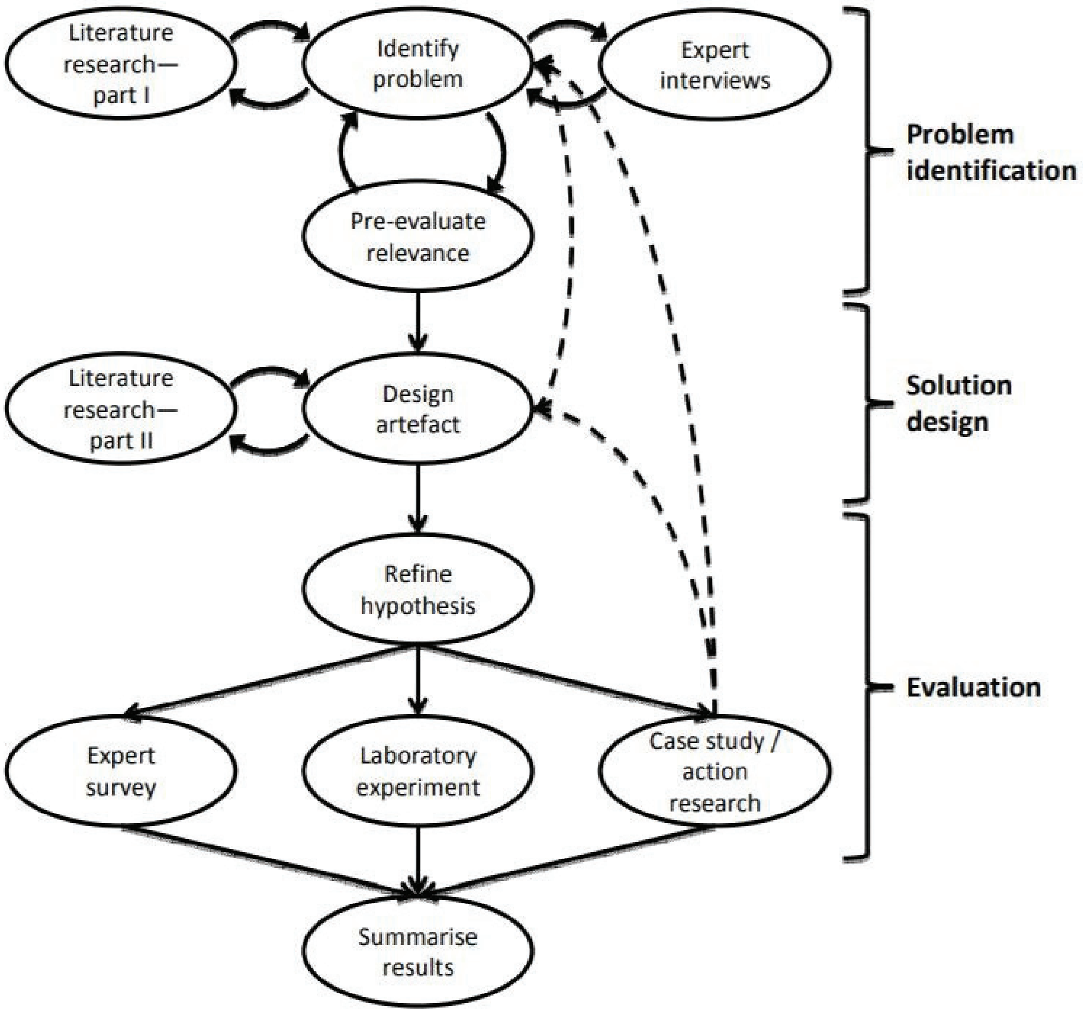

Figure 3.1. DSR process[40].

In this thesis, we adapted the process shown in Figure 3.1 as our research methodology (see Figure 3.2). The following sections describe the research activities adapted from DSR steps.

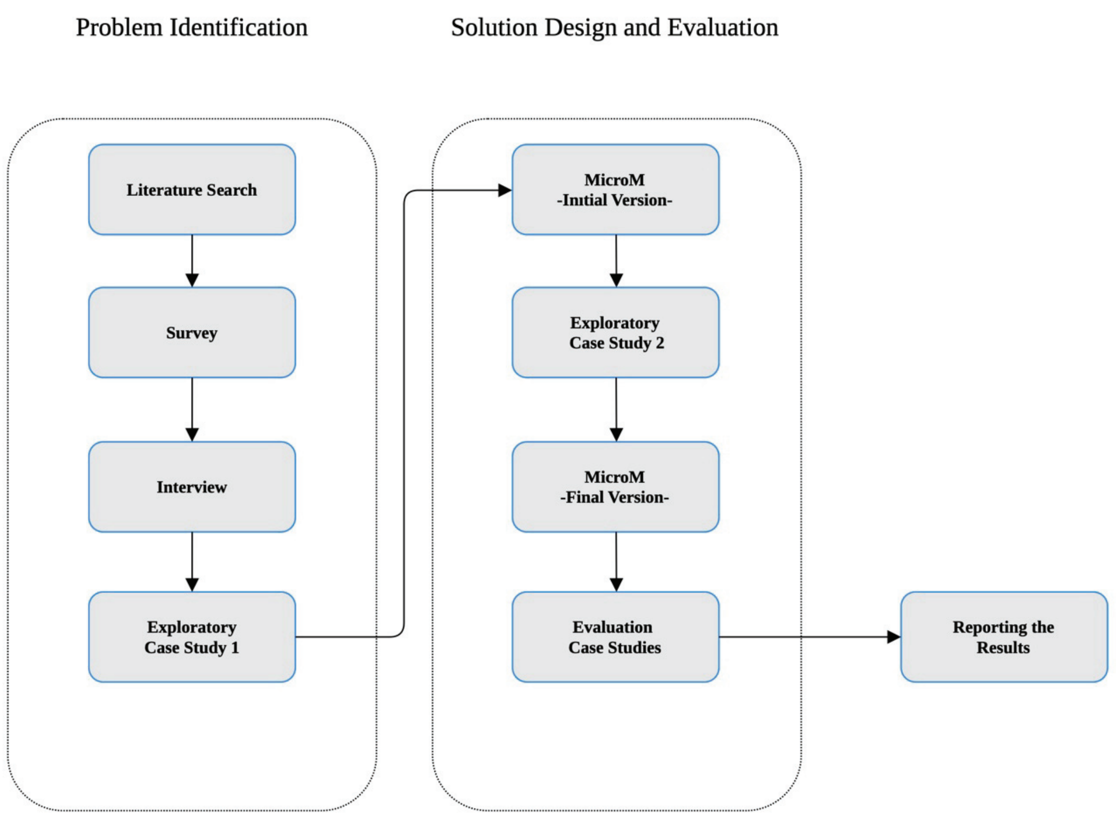

Figure 3.2. Adapted DSR process.

3.1. Problem Identification

The problem identification phase includes steps such as literature review, expert interviews, and pre-evaluation of the problem's relevance.[40] Understanding the current situation is crucial for identifying improvement opportunities.[85]

As the first step of this phase, we conducted a literature review on size measurement in microservices-based projects. The literature review showed a lack of studies on size measurement and effort estimation in MSSAs. Subsequently, we conducted a survey to identify size measurement and effort estimation methods used in microservices-based projects in the industry, as well as the challenges organizations face. We designed the survey based on the guidelines proposed by Shull et al. [86] and Linekar et al.[87].

The survey offered valuable insights to software organizations looking to adapt their culture to MSSAs. However, we noted that the questionnaire-based format limited our ability to fully capture the in-depth perspectives of practitioners, especially in a domain undergoing significant transformation. Therefore, we performed a follow-up structured interview with the practitioners who develop MS-based solutions. The survey and interview results showed that organizations encounter difficulties implementing objective size measurement in MS-based projects, as no widely accepted standard method is utilized in the industry for these projects.

We concluded that utilizing events as a size measurement in MSSAs could be relevant since events represent the base components of microservices. Consequently, we conducted an exploratory case study to assess the accuracy and relevance of event-based size measurement in microservices. In this study, we evaluated the success and suitability of the Event Point[51] method, an event-based size measurement approach identified in our literature review, in MS-based projects. We developed effort estimation models using both the Event Point[51] and COSMIC[34] size measurement methods. The results indicated a better correlation between effort and event-based size than COSMIC size. The following section outlines the details of the solution design phase that we performed based on all these findings.

3.2. Solution Design

Offermann et al.[40] recommend conducting artifact design and literature search as part of the solution design process. Additionally, Hevner et al.[39] and Offermann et al.[40] highlight the iterative and incremental nature of the design process. We adapted an iterative and incremental approach while developing the MicroM size measurement method within these frameworks. During the problem identification phase, we found a correlation between using events, base components of microservices, as a size unit and effort. Based on our literature search, we developed the first version of the MicroM method, considering the characteristics of microservices and the shortcomings of existing FSM methods. We evaluated this method's accuracy and architectural alignment through an exploratory case study. Following the findings from this case study, we improved our method and developed the final version of the MicroM size measurement method. Lastly, we evaluated our proposed size measurement method, which is described in the following section.

3.3. Evaluation

Offermann et al.[40] suggest using expert judgment, lab experiments, or case studies as evaluation methods in the DSR process. We selected the case study method[88] to evaluate our model. The method is also recommended by Hevner et al.[39] as an observational evaluation method for DSR. A case study is an empirical research method that rigorously examines a contemporary phenomenon within a real-world context, where the boundaries between the phenomenon and its context may be unclear.[88] The case study is based on observation and allows researchers to concentrate intensely, holistically, and with a real-world perspective on a phenomenon.[85,88] One additional advantage of case studies over experimental or survey-based research is their ability to explain real-life situations, including their complexities.[89]

Case study research is frequently applied in disciplines such as sociology, psychology, and political science, as well as practical fields like accounting, healthcare, and software engineering.[88] The interdisciplinary nature of the software engineering field, bringing together disciplines where case studies are already used, makes it naturally suited for software engineering.[90] Moreover, by helping avoid scaling issues, case studies are particularly well-suited for evaluating software engineering methods and tools in industrial environments. In this context, a "case" in software engineering can be a project, an organization, a process, or a technology.[90]

A case study can be conducted on any subject and increases understanding of the phenomenon being studied.[88,91] Therefore, it is possible to apply case studies for exploratory purposes, to understand and explain a phenomenon, or to build a theory both prospectively and retrospectively.[92] Additionally, Wohlin et al.[85] note that case studies are conducted as a comparative research strategy where it is possible to compare the results of multiple methods.

Yin[88] classifies case studies into three categories: exploratory, descriptive, and explanatory case studies. Exploratory case studies are conducted to investigate a phenomenon in the data that serves as an area of interest; descriptive case studies are conducted to describe the natural phenomenon that emerges from the data; and explanatory case studies are conducted through surface and in-depth research to explain the phenomenon in the data.

Case studies can be classified as single or multiple, depending on how they are applied.[88] By allowing for replication, multiple case studies create a more robust research framework, enhance the reliability of findings, and provide greater validity to the research outcomes.[88,89,93,94]

In this thesis, we planned to perform an exploratory multiple case study method to provide the first discovery of phenomena to build theories and develop hypotheses.[93] Since this thesis aims to define a new software size measurement method for MSSAs, we could explore the strengths and weaknesses of current size measurement methods in MS-based projects. We adopted multiple case studies process, consisting of six main steps: planning, designing, preparing, collecting, analyzing, and sharing.[88]

We used quantitative analysis techniques, including descriptive statistics, correlation analysis, predictive modeling, and hypothesis testing, to evaluate the case studies. We applied regression analysis to estimate the effort to assess the method empirically. Regression analysis, a technique for examining relationships between variables, is commonly used for data description, parameter estimation, and prediction.[95,96] It is also widely used for effort estimation purposes in the literature.[31,32,97,98] There are two types of regression based on the number of explanatory variables: simple and multiple regression.[95] In this thesis, to estimate effort, we used simple linear regression if the size measurement method provided a single dimension of size and multiple linear regression if it provided multiple dimensions.

In regression analysis, R-squared (coefficient of determination) is used to measure the strength of the relationship between the regression model and the dependent variable. Humphrey[99] proposes the following criteria for correlation relationships in software engineering for planning purposes:

- $0.9 \leq$ R-squared; a predictive relationship
- $0.7 \leq$ R-squared < 0.9; a strong relationship
- $0.5 \leq$ R-squared< 0.7; an adequate relationship
- R-squared < 0.5; not reliable for planning

We used widely recognized metrics from the literature to evaluate the prediction accuracy of the regression-based effort estimation models we developed: (1) Magnitude of Relative Error (MRE), (2) Mean Magnitude of Relative Error (MMRE), (3) Median Magnitude of Relative Error (MdMRE), and (4) Percentage Relative Error Deviation within X (PRED (X)).[90] The MRE metric, used to evaluate prediction accuracy, forms the basis for other metrics.

$$\mathrm{MRE} = \left|\frac{\text{Actual Effort}-\text{Estimated Effort}}{\text{Actual Effort}}\right| \qquad (1)$$

MMRE is calculated by taking the average of the MRE values. The value of n represents the number of observations. In the literature, different threshold values have been proposed for MMRE.[100,101] According to Hastings and Sajeev[101], models with an MMRE less than 0.20 are considered predictive, those with an MMRE between 0.20 and 0.50 are deemed acceptable, and models with an MMRE greater than 0.50 are regarded as not acceptable.

$$\mathrm{MMRE} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{MRE}_i \qquad (2)$$

The MMRE metric is frequently used in the literature. However, it is criticized for being sensitive to high MRE values present among the data.[102–106] In these cases, MdMRE, which is less affected by extreme values, can be used.[107]

In the equation shown for PRED(X), "n" represents the total number of observations, while "k" indicates the number of observations with an MRE value less than "x". In the literature, PRED(0.25) and PRED(0.30) are recognized as the most commonly used PRED values.[108,109] MacDonell and Gray[110] suggest that a model with a PRED(30) accuracy of 60% can be considered a good model.

$$\mathrm{PRED}(x) = \frac{k}{n} \qquad (4)$$

3.4. Summary of the Chapter

In this section, we describe the research methodology applied to our thesis. We adapted DSR to develop a novel software size measurement model for MSSAs. The DSR process consists of three iterative phases: (1) Problem Identification, (2) Solution Design, and (3) Evaluation. We explained how we adapted these three phases proposed by Offermann et al.[40] in our thesis. The following sections describe the execution of the adapted DSR research methodology.

CHAPTER 4

DEVELOPMENT OF THE SIZE MEASUREMENT METHOD

Following the DSR stages, after the literature review, we conducted a survey, interviews, and an exploratory case study during the problem identification phase before finalizing the MicroM size measurement method. In the solution design phase, we first developed the initial version of our MicroM size measurement method and then evaluated its success in the exploratory case study. The details of all these steps are explained in the following sections.

4.1. Survey

The literature review showed a lack of studies on size measurement and effort estimation in MSSAs. To determine the practices organizations, prefer for size measurement and effort estimation in MS-based projects, which have been present in the software world for about ten years, we conducted a survey.[38] The details of the survey are provided in the following sections.

4.1.1. Research Method

We used a survey method to understand how organizations handle MSSAs. We opted for an anonymous online survey for three main reasons. First, it allowed us to reach a large number of organizations worldwide quickly. Second, anonymity encouraged companies and employees to share information about their development processes without hesitation. Finally, we wanted the collected data to be easy to categorize and analyze. We chose Google Forms as our survey platform because it's user-friendly and free, exports result in XLSX format and presents data clearly.

One common issue with surveys is unclear questions.[86] To address this, we conducted a pilot survey to identify any ambiguous questions. This pilot survey also helped us gauge the time it took for a first-time respondent to complete the survey and allowed us to collect feedback. We made some adjustments before launching the survey based on the pilot feedback. Following these adjustments, we submitted the survey to the Human Subjects Ethics Committee (HSEC) at Izmir Institute of Technology (IZTECH) for approval.

4.1.1.1. Goal and Research Questions

The primary goal of this survey is to understand the practices organizations follow when working with MSSAs. From the results, we aim to identify which practices are widely adapted, which are less common, and what their limitations are. Thus, we formulated the research question for this study as follows:

- *What are the industrial common practices and challenges encountered in software size measurement and effort estimation in MS-based projects?*

4.1.1.2. Sampling Method

We employed accidental non-probabilistic sampling in the survey.[87] Our target participants were professionals in the software field with experience in MSSAs. We primarily relied on personal and company contacts, forums, mailing lists, and LinkedIn to reach these individuals. We also encouraged participants to share the survey with their colleagues as long as no more than five individuals from the same company participated. As the survey was widely shared, we were unable to track who completed it due to its anonymity. However, we categorized participants based on whether they had experience with MSSAs to facilitate data analysis.

4.1.1.3. Designing Survey Questions

Our survey is divided into four sections. The first section provides an overview of the survey and asks participants to give their consent. The second section focuses on demographics, collecting general information about the participants, their organization, their overall experience, and whether they have experience with microservices. If a participant indicates they have no experience with microservices, the survey ends at this point. The third section is dedicated to experience with microservices, and the fourth section gathers information about the size measurement and effort estimation techniques used by participants in MSSAs. Survey questions are presented in Table 4.1.

Table 4.1. Survey questions.

Section	**Survey Question**	**Type of Answers**			
		Single Answer	**Multiple Answer**	**Free Text**	**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 architecture?	X			
3	How long (in years) have you been working with microservices?			X	
	What is the domain of your current microservice-based project?	X			
	What is the type of your current microservice-based project?	X			
	Which software development methodology do you follow when working with microservices?	X			

(cont. on next page)

Table 4.1 (cont.). Survey questions.

Section	**Survey Question**	**Type of Answers**			
		Single Answer	**Multiple Answer**	**Free Text**	**Likert Scale**
	Which software development methodology do you follow when working with microservices?	X			
	What is the team on your current microservice-based project?			X	
	How many microservice-based projects are running concurrently in your organization?			X	
4	Which software size measurement method do you use in your microservice-based projects?	X			
	How do you perform effort estimation in your microservice-based projects?	X			
	Which tasks are included in effort estimation in your microservice-based projects?		X		
	Have you been able to predict effort more precisely when utilizing microservice architectures?				X
	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			

4.1.1.4. Survey Piloting and Execution

A pilot test was conducted with three individuals from different companies and roles to clarify and avoid misunderstandings. The survey's completion time was calculated, and the following adjustments were made based on the given feedback.

- We removed the requirement to log in with a Gmail account.
- We added a control question to verify whether the participant had experience with microservices, with the survey ending for those who did not.

- We changed the questions regarding the participant's experience from monthly to yearly intervals.
- We replaced some open-ended questions with multiple-choice options, including an "Other" option for flexibility.

4.1.1.5. Criteria for Validation

We validated the survey primarily through the seventh question (Q7), which asked participants if they had experience with MSSAs. Although we emphasized "microservice" in both the survey title and participant invitations, some individuals without this experience still completed the survey. To address this, we included a final question in Section 2 to confirm their experience and end the survey early if they selected "No." After collecting the responses, we excluded participants without microservice experience from the data analysis.

4.1.2. Survey Results

We summarize the survey results from 67 participants as follows:

4.1.2.1. Participant Demographics

The 67 participants in the survey were from nine countries across four continents: Turkey, Australia, Germany, Italy, China, the Netherlands, Southern Cyprus, the United Kingdom, and the USA (see Figure 4.1a). Most participants (85%) were from Europe, 12% from America, and 3% from Asia and Australia. In terms of education, participants had undergraduate degrees in fields such as Computer Engineering, Computer Science, Software Engineering, Information Systems, Electrical and Electronics Engineering, Industrial Engineering, and other disciplines like Mechanical/Mechatronics Engineering,

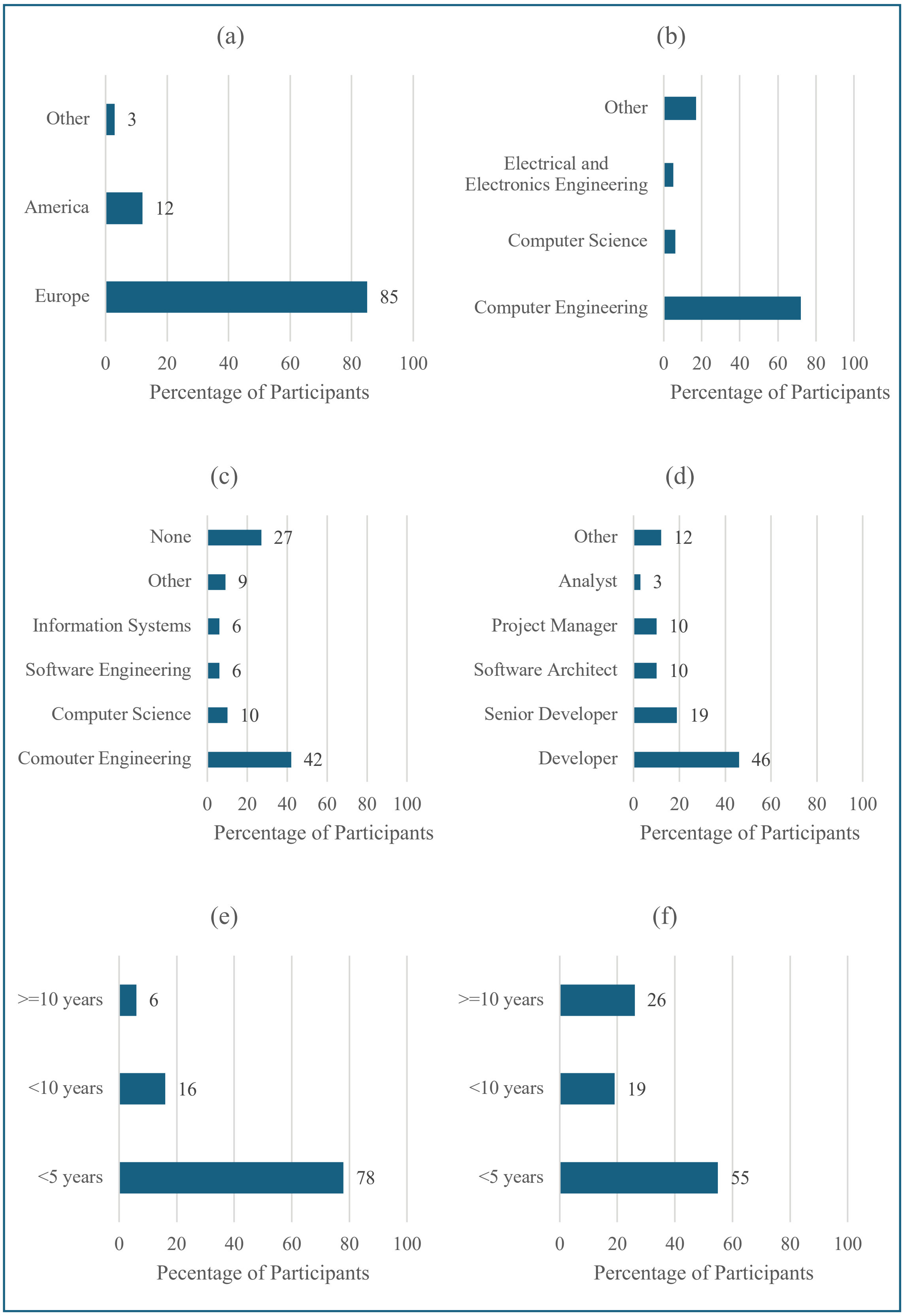

Figure 4.1. Survey responses regarding demographics - (a) organizations' geographical distribution, (b) undergraduate studies, (c) graduate studies, (d) role in the organization, (e) experience in the current role, and (f) experience in the software field.

Mathematics Engineering, Business, Economics, and Industrial Engineering (see Figure 4.1b). Nearly 73% of participants also had graduate degrees in areas like Computer Engineering, Computer Science, Information Systems, and Software Engineering, as well as in Business, Cyber Security, Sustainable Development, Electrical and Electronics Engineering, Industrial Engineering, and Management Science (see Figure 4.1c).

Participants held various roles within their organizations (see Figure 4.1d). Most participants worked as developers (46%) or senior developers (19%), with 10% as software architects, 10% as project managers, and 3% as analysts. Their experience in these roles ranged from 1 to 15 years, with an average of 2.95 years (see Figure 4.1e). Most had less than five years of experience in their current role. Participants' total experience in the software field ranged from 1 to 30 years, averaging 6.5 years (see Figure 4.1f). Most participants had fewer than five years of experience in their current role.

To analyze the results, we divided participants based on whether they had experience with MSSAs, using a question at the end of the survey to determine this. After completing the survey, participants without microservice experience were excluded from the analysis. Of the 67 participants, 45 (67%) had experience with microservices.

4.1.2.2. Experience with Microservices

The participants had an average of 2.4 years of experience with microservices, indicating that most had less than five years of experience. (see Figure 4.2a). Their most recent MS-based projects spanned 20 different domains, with finance, telecom, mobile software, web-based productivity apps, e-commerce, and entertainment being the most common.

In examining their last MS-based project, we found that 47% developed a new MS-based system, 36% reimplemented a monolithic system as microservices, 16% reengineered parts of an existing system to work with a monolith, and 2% replatformed an already MS-based system (see Figure 4.2b). Most participants (53%) used Scrum as their software development methodology, 24% favored Kanban, and 9% followed Extreme Programming (see Figure 4.2c).

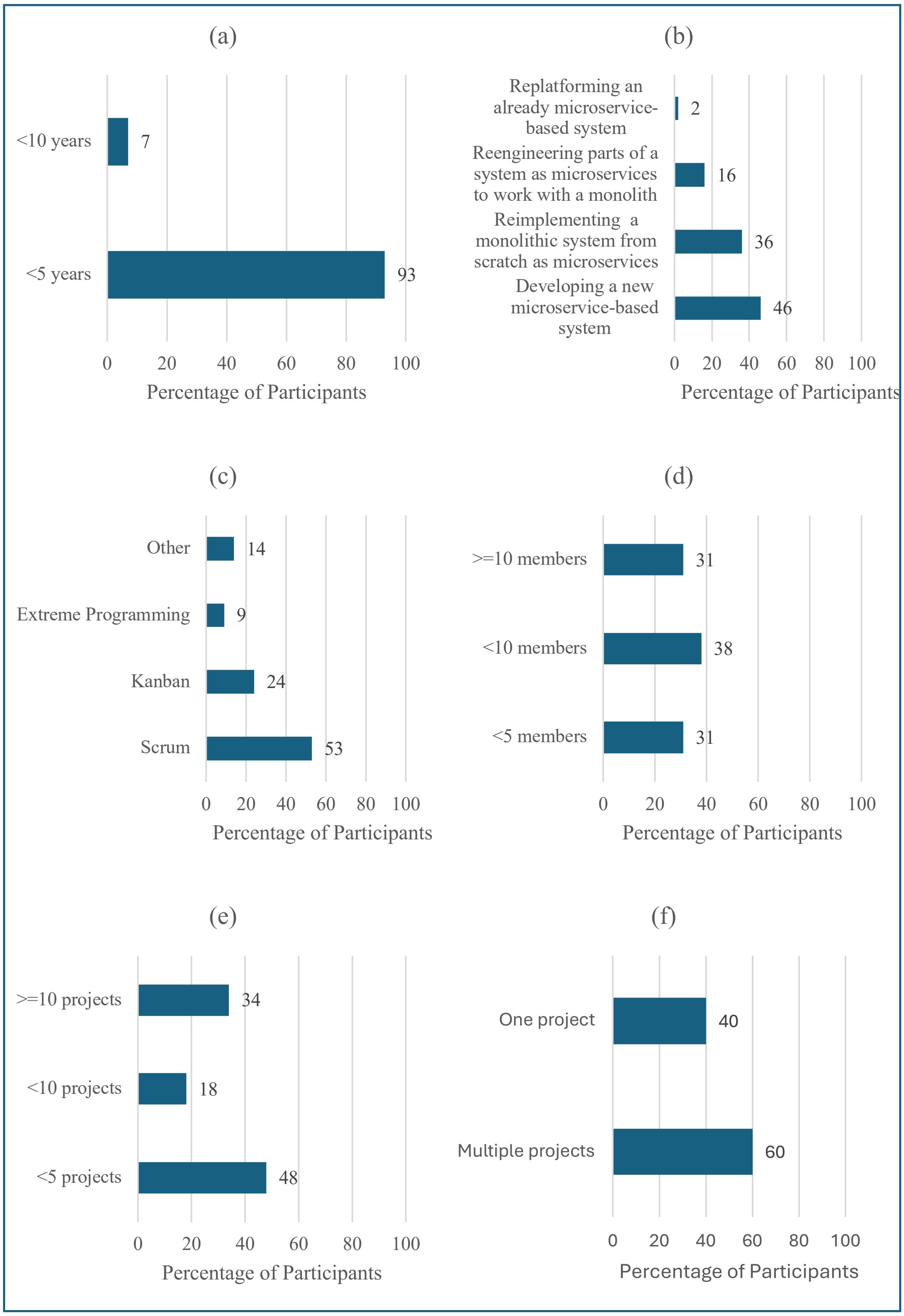

Figure 4.2. Survey responses regarding experience - (a) experience with microservices, (b) the type of the current MS-based project, (c) applied agile methodology, (d) average team size, (e) the number of projects in the organization, and (f) the number of projects teams usually work on.

The average team size for their MS-based projects was 9.3, indicating that most teams had fewer than ten members (see Figure 4.2d). Participants reported a median of 5 projects concurrently running MS-based projects within their organizations (see Figure 4.2e). Lastly, most participants reported being involved in multiple projects simultaneously (see Figure 4.2f).

4.1.2.3. Size Measurement and Effort Estimation

Most participants (38%) use story points to measure software size in MS-based projects. 18% use T-shirt sizing, 9% use source lines of code, and 9% use use-case points. Notably, 22% do not use any size measurement method (see Figure 4.3a).

For effort estimation, planning poker is the most common approach, utilized by 31% of organizations. Twenty-four percent (24%) rely on expert judgment, 16% use ad-hoc methods, and 11% apply parametric techniques (see Figure 4.3b). Effort estimation includes various tasks: 91% include development effort, 76% include testing effort, 58% include analysis effort, 51% include design effort, and 38% include operational effort. In 11% of organizations, effort is not estimated (see Figure 4.3c).

Most participants believe that effort estimation is more accurate with microservice architectures: 9% strongly agree, and 42% agree with this statement. Meanwhile, 29% feel microservice architectures do not affect precision (see Figure 4.3d). Regarding the efficiency of effort estimation, 51% see no change, 24% find it more efficient, and 13% find it very efficient when using microservices (see Figure 4.3e).

Most participants track their actual individual effort daily (31%) or weekly (37%), while 22% do not record their individual effort (see Figure 4.3f). This percentage matches the proportion of participants who do not use any size measurement method.

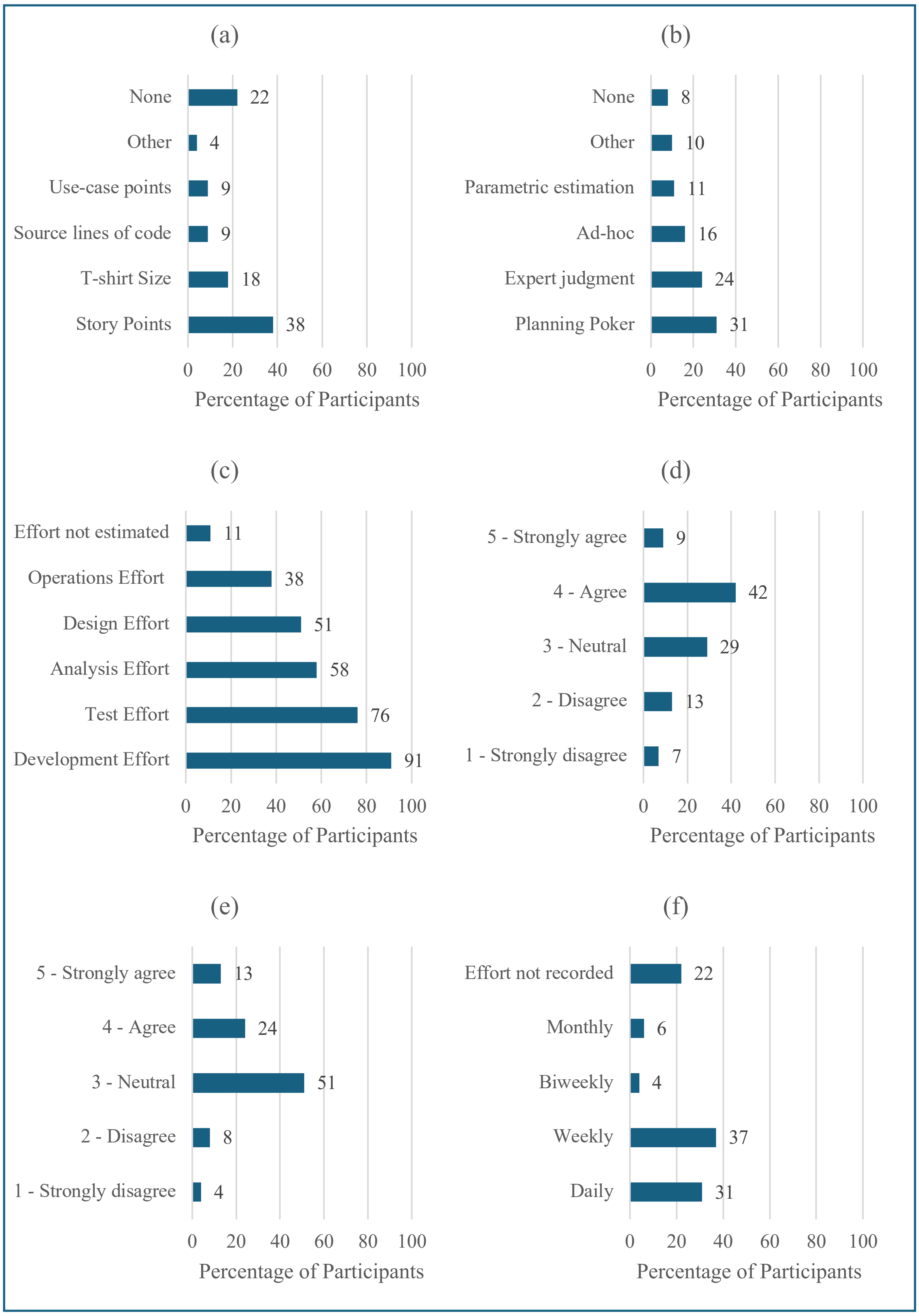

Figure 4.3. Survey responses regarding size measurement and effort estimation - (a) software size measurement methods, (b) effort estimation techniques, (c*) included tasks in effort estimation, (d) precision of the effort prediction, (e) changes in the efficiency of the effort estimation, and (f) actual individual effort record frequency (*: multiple answers).

4.1.3. Implications from the Survey

The survey results showed that organizations generally rely on subjective size measurement and effort estimation methods in their MS-based projects. However, it is important to investigate why they choose these methods: Do they face challenges when using existing objective size measurement methods in MS-based projects, or do they find subjective methods to be more effective overall? Secondly, participants generally indicated that effort estimation is more accurate in MS-based projects. We believe the independent design of microservices in these projects may lead to more consistent effort estimation than complex monolithic systems. The survey results revealed that a detailed analysis of these findings is necessary.

4.2. Interview

The survey offered valuable insights to software organizations looking to adapt their culture to MSSAs. However, we noted that the questionnaire-based format limited our ability to fully capture the in-depth perspectives of practitioners, especially in a domain undergoing significant transformation. Therefore, we performed a structured interview to explore the organizational choices and the challenges in size measurement, and effort estimation[37]. The details of the interview are provided in the following sections.

4.2.1. Research Method

We conducted structured interviews to investigate how organizations apply size measurement and effort estimation in projects that adopt the MSSA paradigm. The questionnaire-based survey is another widely used method for gathering information in software engineering. Each approach has its pros and cons. Surveys allow researchers to quickly reach a larger number of practitioners, but in transformation projects, they can lead to misunderstandings and make it harder to identify innovative practices. Structured interviews, though more time-consuming and labor-intensive, offer key advantages. They facilitate in-depth discussions, reduce the risk of misinterpreting questions, and provide researchers with a deeper understanding of the topic.

4.2.1.1. Goal and Research Questions

This work aims to provide insights into how software size measurement and effort estimation are carried out in MS-based projects. By gathering information from industry practitioners, we seek to identify the commonly used methods, techniques, and tools, along with their advantages and limitations. Additionally, we aim to uncover the typical challenges faced when transitioning from monolithic to MSSAs, particularly in size measurement and effort estimation approaches. Thus, we formulated the research question for this study as follows:

- *What are the industrial common practices and challenges encountered in software size measurement and effort estimation in MS-based projects?*

4.2.1.2. Target Audience

Our target participants were software professionals involved in MS-based projects using agile methodologies. We primarily relied on personal or company contacts and LinkedIn to reach these individuals. During the selection process, we ensured that the participants represented companies from various countries and industries to capture diverse perspectives.

4.2.1.3. Interview Design and Execution

In designing the interview process, we ensured that all questions were carefully aligned with the research scope and objectives, with the intent of eliciting information relevant to addressing the research questions.

The interview questions were organized into four sections, each focusing on a specific area of interest. The first section was designed to inform participants about the objectives of the interview and how the collected data would be used. Participants were informed that their participation was entirely voluntary, with the option to withdraw at any time. The following two sections covered demographic information and general details about the participants and their respective organizations. The last focused on size measurement and effort estimation. The interview included various types of questions, such as single-answer multiple-choice, multiple-answer multiple-choice, and Likert scale-based questions.

Most questions included a free-text option, allowing participants to provide additional details or explanations or to specify an option not listed. This design aimed to collect as much relevant information and insight as possible. A complete list of interview questions can be found in Table 4.2.

Table 4.2. Interview questions.

Section	**Survey Question**	**Type of Answers**			
		Single Answer	**Multiple Answer**	**Free Text**	**Likert Scale**
2	What is the origin country of your organization?			X	
	What is the field of your formal undergraduate education?	X		X	
	Do you have a graduate degree? If yes, what is the field of your latest formal graduate (MSc. or PhD.) education?	X		X	
	What is your role in your organization?	X		X	
	How long (in years) have you been working in a software engineering-related field			X	
	How long (in years) have you been working i in your current role?			X	
	How long (in years) has your organization been working in the software field?			X	
3	What is the domain of your current company/team?	X		X	
	How long (in years) has your organization been using an Agile methodology like Scrum?			X	
	Which agile methodology do you use?		X	X	

(cont. on the next page)

Table 4.2 (cont.). Interview questions.

Section	**Survey Question**	**Type of Answers**			
		Single Answer	**Multiple Answer**	**Free Text**	**Likert Scale**
3	How long (in years) has your organization been working with MSSAs?			X	
	What is the average team size for your current company/team?			X	
	In your organization, do team members generally work on one project?	X			
4	How do you perform effort estimation in microservice-based projects?		X	X	
	Do you measure the size of the projects to estimate the effort? If yes, which method?		X	X	
	Which tasks are included in effort estimation?		X	X	
	What is the smallest unit that you record/estimate the effort?		X	X	
	How frequently is actual individual effort recorded in your organization?	X		X	
	Do you use any tool for effort estimation/size measurement?	X		X	
	Have you observed challenges during the effort estimation process when utilizing MSSAs? What are they?			X	X
	Have you observed any change in the precision of estimated effort when utilizing MSSAs?			X	X

The interviews were conducted remotely via Microsoft Teams, with each session lasting approximately 50 minutes. At the outset, the objectives of the interview were clearly communicated, and the intended use of the data was explained. Following this, a series of structured questions was presented to the participants, with the questions displayed on-screen. One interviewer read and clarified each question while all interviewers took detailed notes. The participants were able to view the notes in real-time, which helped to minimize potential misunderstandings of their responses.

Each interview involved at least two, and occasionally three, interviewers who actively clarified questions and sought further elaboration where necessary. Additionally, with the participant's consent, the interviews were recorded to facilitate the review and resolution of any ambiguities in the responses.

4.2.1.4. Interview Piloting and Execution

After drafting the initial set of interview questions, the interview was reviewed by three experts in the field. Based on their feedback, we revised the questions to improve clarity and alignment with our research objectives. We conducted a pilot interview to further reduce the risk of misunderstandings and ambiguities and ensure the interview design effectively met our goals. Following discussions between the participants and interviewers, additional revisions were made.

Once the necessary updates were finalized, we scheduled interviews with the selected participants. A total of 21 professionals with experience in MS-based projects utilizing agile methodologies were interviewed. Upon completing the interviews, the interviewers collectively analyzed the responses and generated detailed reports based on the findings.

4.2.1.5. Criteria for Validation

A key criterion for participant selection was their experience in the microservice domain. Preselected candidates not meeting this requirement were excluded from the pool before initial contact. During preliminary discussions, some candidates indicated they lacked the necessary expertise to adequately respond to the interview questions and were subsequently removed from consideration.

Following the completion of the interviews, the responses were subjected to a thorough analysis. During this phase, it was identified that several participants had misunderstood a question regarding effort estimation, leading to unreliable data. As a result, this question was excluded from the final analysis, and the corresponding responses were disregarded. Furthermore, it became evident that one participant lacked sufficient domain knowledge, and their responses were deemed unreliable. Consequently, this participant's interview data were excluded from the study. Therefore, the final analysis is based on the interview results from 20 qualified participants.

4.2.2. Interview Results

In this section, we present a summary of the interview results from the 20 participants, organized into three main sections: (1) demographics, (2) organizational information, (3) size measurement and effort estimation.

4.2.2.1. Participant Demographics

In this section, we aimed to collect general information about the participants in the study. The sample consisted of 20 individuals working on MS-based projects from seven countries. Figure 4.4a illustrates the distribution of the countries where the participants' organizations are located.

Most participants hold undergraduate degrees in computer engineering, with 90% graduating in this field, while 5% have degrees in software engineering and another 5% in computer science (see Figure 4.4b). Additionally, 60% of participants do not possess a graduate degree. Among those who do, 15% have a graduate degree in computer engineering, 10% in software engineering, 10% hold a Master of Business Administration (MBA), and 5% have a graduate degree in information systems (see Figure 4.4c).

Participants occupy various roles within their organizations, including software engineer, senior software engineer, developer, senior developer, and positions such as chief executive officer and project manager. Most participants work as software engineers, with the distribution of roles illustrated in Figure 4.4d.

Experience in software engineering-related fields among participants ranges from 2 to 29 years. Notably, 45% of participants have over ten years of experience, while 30% have between 5 and 10 years, and 25% have less than five years of experience (see Figure 4.4e). Participants' experience varies from 1 to 18 years in their current roles. Specifically, 15% have more than ten years of experience in their current positions, 20% have between 5 and 10 years, and 65% have less than five years (see Figure 4.4f).

The organizations represented by participants also vary in their experience within the software field, with durations ranging from 3 to 47 years. Ten percent of the organizations have been established for over 40 years, while 5% have been in operation for 30 to 40 years, 35% for 20 to 30 years, 30% for 10 to 20 years, and 20% for less than ten years (see Figure 4.4g).

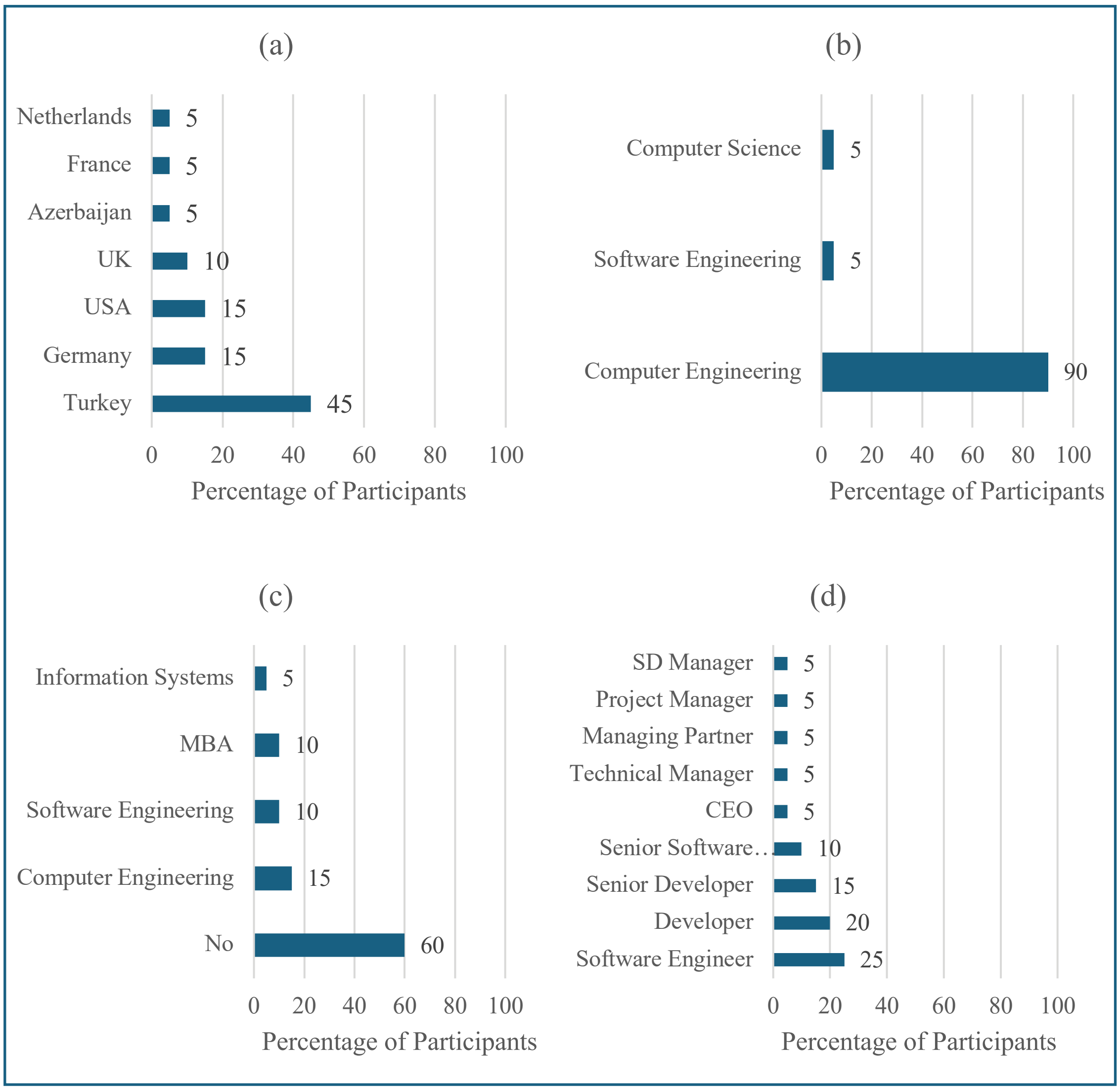

Figure 4.4. Interview responses regarding demographics - (a) the origin country of the organization, (b) undergraduate studies, (c) graduate studies, (d) current role in the organization, (e) years of experience the organization has in the software field, (f) years of experience in the current role, (g) years of experience in the software field (cont. on next page).

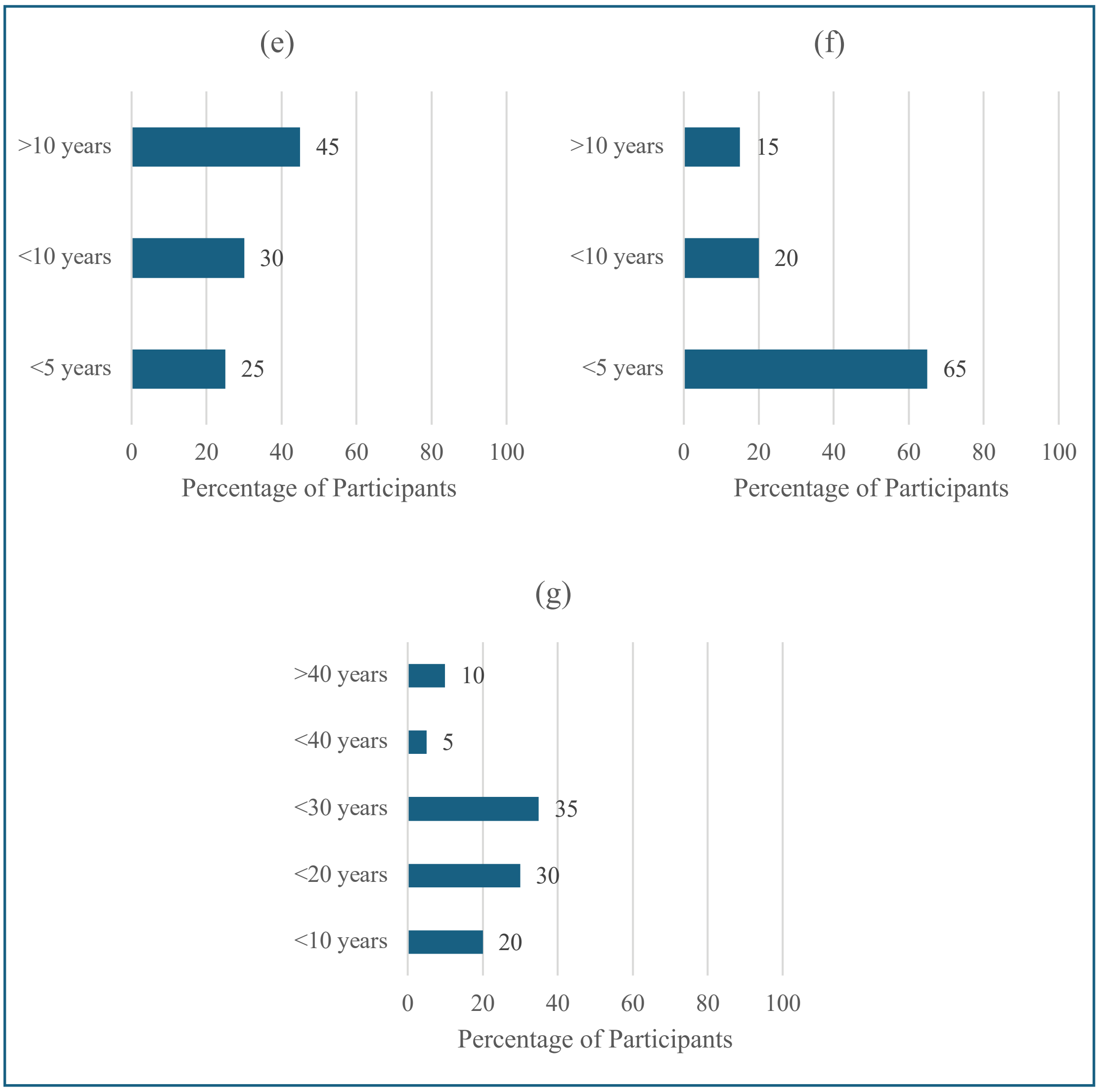

Figure 4.4 (cont.)

4.2.2.2. Organizational Information

We asked participants to respond to questions regarding their organizations to better understand their expertise in agile methodologies and MSSAs. Our findings indicate that 30% of participants are from the finance sector (see Figure 4.4a), followed by software consulting at 25% and e-commerce at 10%. The remaining participants identified with distinct categories, such as military software (5%) and other sectors (30%), which include car rental and telecommunications.

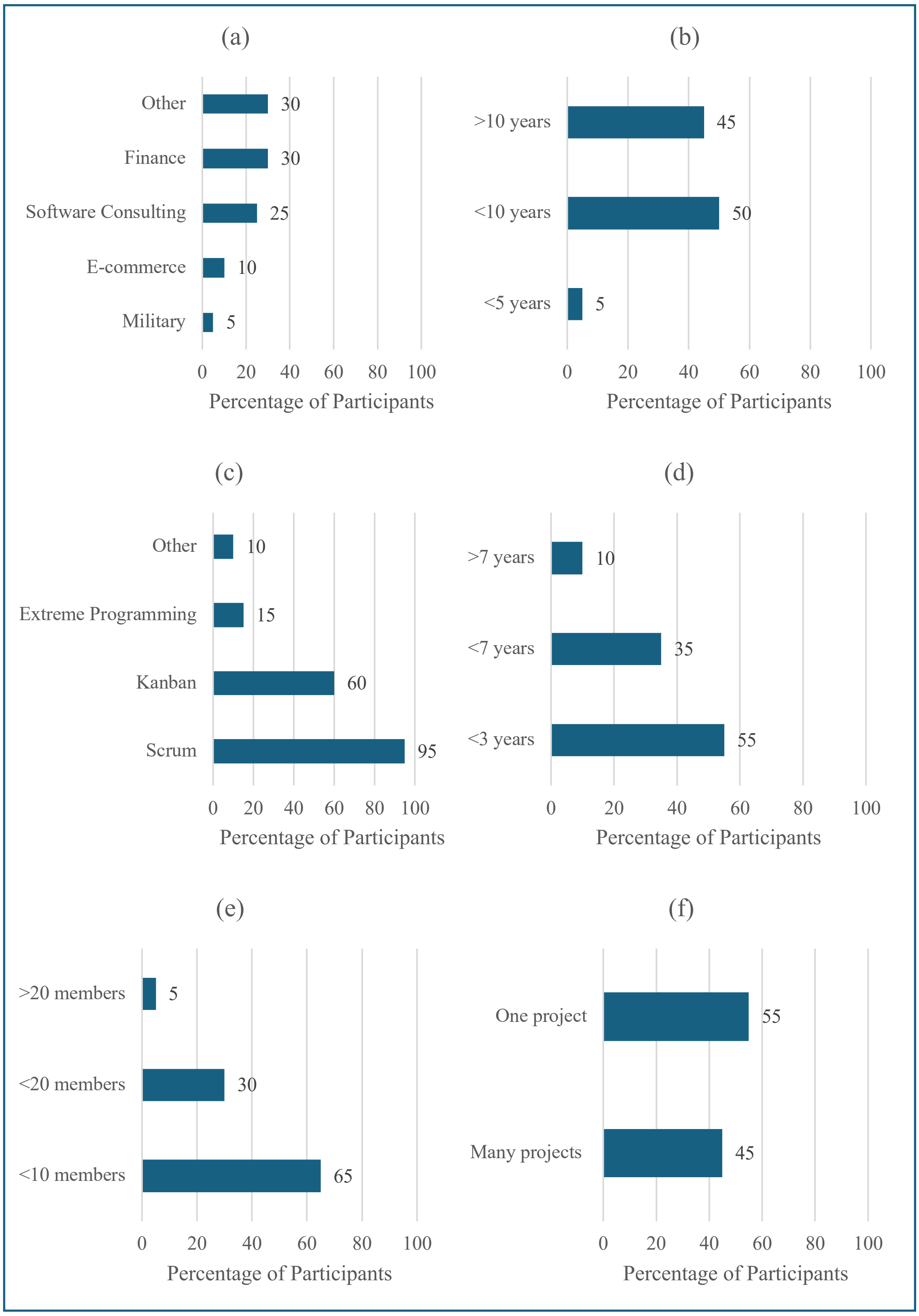

Figure 4.5. Interview responses regarding experience - (a) domain of organization, (b) years of experience the organization has in using an agile methodology, (c*) applied agile methodology, (d) years of experience the organization has with microservice architecture, (e) average team size, (f) the number of projects teams usually work on (*: multiple answers).

Regarding experience with agile methodologies (see Figure 4.5b), half of the participants (50%) have utilized such methodologies for between 5 and 10 years, while 45% have done so for less than five years. Only 5% reported using agile methodologies for more than ten years. To gain insights into industry trends, we inquired about the specific agile methodologies employed in participants' current organizations, allowing for multiple responses. Notably, the total percentage exceeds 100, as many participants indicated using more than one methodology. According to the responses (see Figure 4.5c), most participants utilize Scrum (95%), followed by Kanban (60%). Only 15% reported using Extreme Programming, while 10% mentioned other methodologies, such as SAFe. An interesting observation is that all participants who reported using Kanban stated that they do so in conjunction with Scrum, with some also mentioning the use of Extreme Programming alongside Kanban, whereas many indicated using Scrum independently.

In response to a question about their experience with MSSAs (see Figure 4.5d), we found that 55% of teams have been utilizing this architecture for less than three years, 35% for between 3 and 7 years, and 10% for more than seven years. Team sizes (see Figure 4.5e) predominantly consist of fewer than ten members, with 65% falling into this category. Only 5% of teams comprised more than 20 members, while 30% had between 10 and 20 members.

In the last question, we examined whether teams were working on a single project or multiple projects simultaneously (see Figure 4.5f). The results revealed a close distribution, with 55% of participants indicating that they are engaged in a single project, while 45% reported working on multiple projects concurrently.

4.2.2.3. Size Measurement and Effort Estimation

We began by asking how organizations approach effort estimation in their MS-based projects (see Figure 4.6a). Most (65%) utilize Planning Poker, and 50% apply Expert Judgment. Notably, many teams combine these two methods. A smaller percentage of organizations use Parametric Estimation (5%), Wide-band Delphi (5%), and Ad-hoc Estimation (5%). Additionally, 10% of participants reported that their organizations do not estimate effort at all.

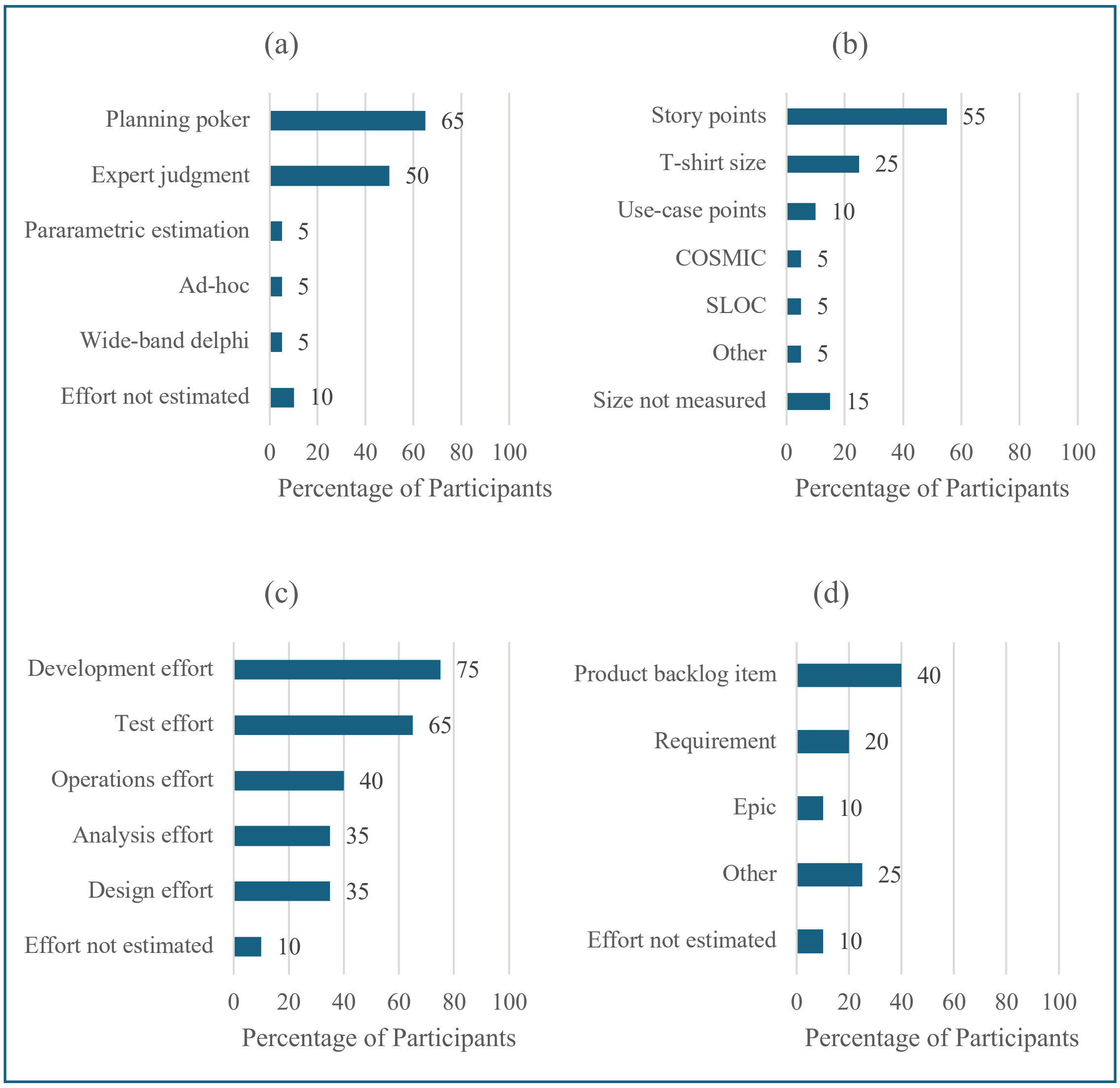

Figure 4.6. Interview responses regarding size measurement and effort estimation - (a*) method used for effort estimation, (b*) method used for size measurement, (c*) tasks included in effort estimation, (d*) smallest unit used in effort estimation, (e) frequency of recording individual work, (f) tool used for measurement or effort estimation, (g) difficulty of effort estimation, (h) precision of effort estimation (*: multiple answers) (cont. on next page).

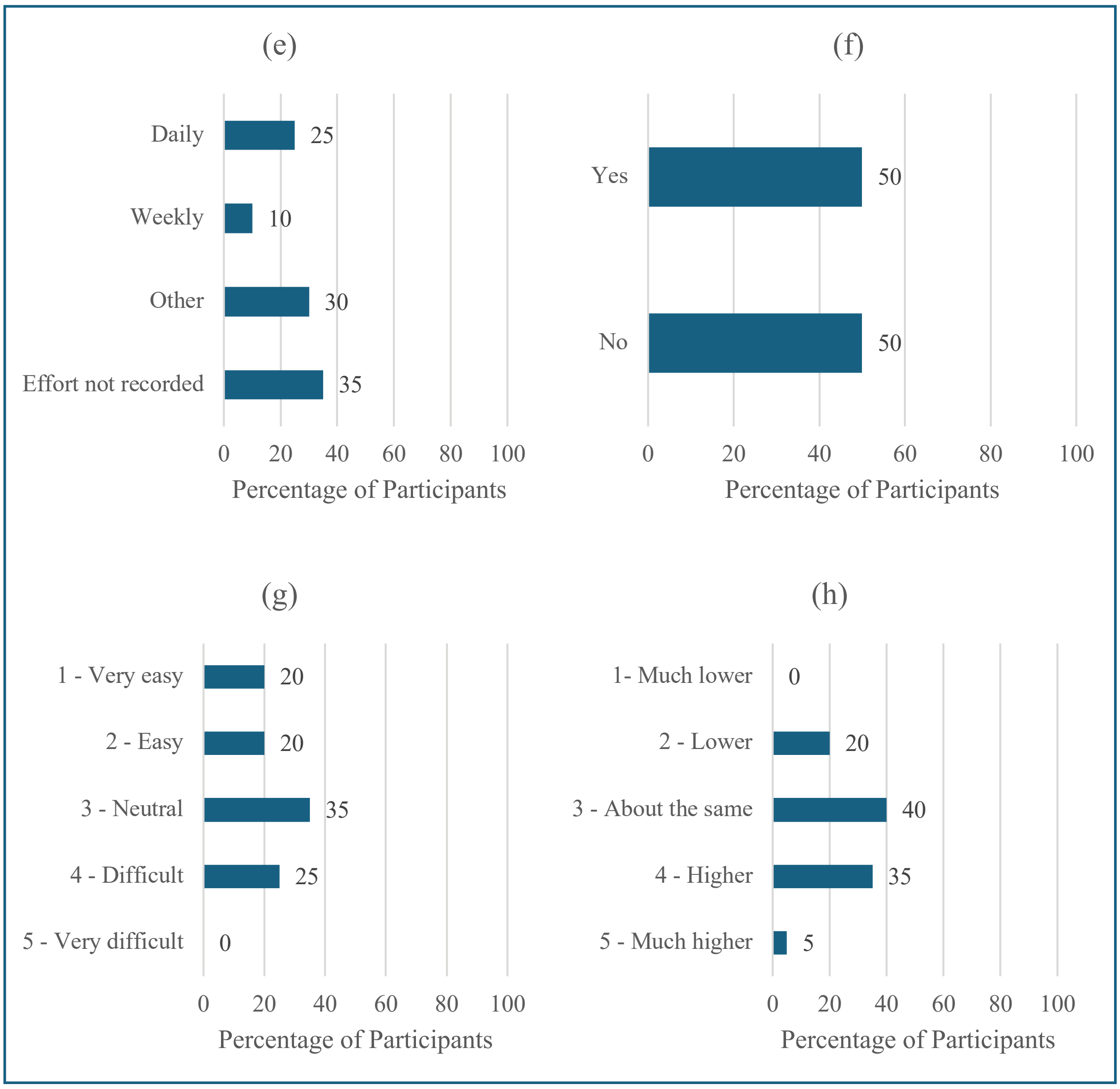

Figure 4.6 (cont.)

Next, we inquired whether organizations measure software size and, if so, which methods they employ (see Figure 4.6b). The findings align closely with those on effort estimation, with most organizations favoring subjective approaches. Specifically, 55% use Story Points, 25% use T-shirt Sizing, and 10% employ Use-case Points. Some organizations combine Story Points and T-shirt Sizing depending on the project. On the formal side, 5% of organizations use COSMIC, and another 5% use Source Lines of Code (SLOC). However, 15% of participants indicated that their organizations do not measure software size.

We then explored which tasks are typically included in effort estimation (see Figure 4.6c). Among participants who perform effort estimation, 75% include Development effort, 65% include Testing effort, 40% include Operations effort, and 35% account for Analysis and Design separately. The results show that most teams incorporate Development, Testing, and Design efforts together in their estimates.

We also asked about the smallest unit used for effort estimation (see Figure 4.6d). A significant number of participants (40%) estimate at the Product Backlog Item (PBI) level, 20% at the Requirement level, and 10% at the Epic level. Additionally, 25% estimate at varying levels, such as User Story, component, or sprint, while 5% do not perform effort estimation at all.

Regarding the frequency of recording actual individual effort (see Figure 4.6e), we found that although most organizations estimate effort, 35% do not record actual individual effort. Among those that do, 25% record effort daily, and 10% do so weekly. In 30% of organizations, effort is recorded in various ways, including hourly with automation, sprint-based, or by tracking only the remaining time to complete tasks.

We then inquired whether any tools are used for size measurement or effort estimation (see Figure 4.6f). Half of the organizations (50%) use tools, with Planning Poker applications being common, while the other 50% prefer not to use any tools.

In the seventh question, we asked about the challenges encountered during effort estimation in MS-based projects (see Figure 4.6g). Thirty-five percent of participants reported no significant changes in the challenges compared to traditional projects. Twenty-five percent found effort estimation in MS-based projects more difficult, whereas 20% found it easy, and another 20% found it very easy. Participants who found the process easy or very easy mentioned that although the migration posed initial difficulties, they became more experienced and successfully completed the transition.

Finally, we asked participants to assess the precision of effort estimation in MS-based projects (see Figure 4.6h). The results mirror the trends observed with challenges. Forty percent of teams reported that the precision of effort estimation was comparable to traditional projects, while 35% observed higher precision, and 5% reported much higher precision. Conversely, 20% of participants noted lower precision. As with challenges, those who achieved higher precision had typically completed the migration process successfully.

4.2.3. Implications from the Interview

In agile development, subjective methods like Planning Poker and Expert Judgment dominate.[15] Our interview findings reflect this, with many organizations combining these two methods—teams estimate using Planning Poker, but management often adjusts the final numbers.

Software size measurement, essential for effort estimation and process improvement, follows a similar trend. Most organizations rely on subjective methods, with Story Points being the most common in MS-based projects. While widespread, these methods have drawbacks such as inconsistent size perceptions and team-specific biases, making them hard to apply across different teams. This reliance on subjective methods highlights the potential need for formal approaches in agile MS-based projects.

Although formal methods like FSM have been successfully applied in agile projects, they are challenging to implement in MSSAs. Methods like COSMIC, which depend on data movement analysis, are difficult to apply to microservices, where events and minimal documentation complicate traditional measurement techniques. This suggests a need for new, microservice-specific effort estimation methods.

Participants highlighted challenges in effort estimation for microservices, particularly due to the dependencies between teams. Despite microservices' independence, inter-service communication creates synchronization issues. Some participants reported that effort estimation became so complex due to these dependencies that they abandoned the practice altogether. However, experienced teams found fewer challenges, emphasizing the importance of team familiarity with the domain and architecture.

Most participants found the precision of estimates similar to that of traditional architectures. Organizations that had completed their migration to microservices and those that struggled with precise estimation, in general, both reported this. However, teams that succeeded in improving precision attributed it to clearly defined microservices. Still, larger projects with multiple interconnected microservices often saw deviations from initial estimates due to the complexity of their systems.

4.3. Exploratory Case Study 1

During our Problem Identification phase, based on the findings from the literature review, survey, and interview, we determined that using events to measure size in MSSAs would be suitable, as events serve as the base components of microservices. Therefore, we conducted an exploratory case study to evaluate the success and suitability of event-based size measurement in microservices. In this study, we estimated effort in an MS-based project using the Event Point method, an event-based size measurement approach proposed by Hacaloglu[51], alongside the COSMIC method[33]. The details of this study are outlined in the following sections.

4.3.1. Case Study Design

This case study aims to identify the challenges organizations face when applying software size measurement and effort estimation practices to MS-based projects in agile software development. Additionally, it seeks to provide recommendations to address these difficulties and improve the overall process. To achieve this, we formulated the following research questions:

- *RQ1. Is "event" relevant as a size measure in MSSAs?*
    - *RQ1.1. How successful are the Event Points and COSMIC methods for size measurement and effort estimation in organizations using agile practices within MSSAs?*
- *RQ2. What are the difficulties encountered in terms of software size measurement and effort estimation practices for MSSAs in organizations utilizing agile methods?*

4.3.1.1. Case Selection Criteria

The key criterion for selecting a project for this case study is that it should be developed using the agile software development method in MSSA. Additionally, the project's requirements should be documented to allow for accurate measurement. Lastly, the effort data corresponding to these measured requirements must be recorded to build the effort estimation model.

4.3.1.2. Description of the Case

The case study focuses on a project designed to help businesses better manage and control their field service operations. Initially targeting the air conditioning sector, the software is expected to expand into other industries in the future. It offers various functionalities in both mobile and web versions, catering to a diverse range of customers with different package options. The project exemplifies modern software design using an event-based architecture.

The organization behind the project specializes in software that helps clients digitize their sales, logistics, and service operations, and it has over 30 years of industry experience. They manage more than 130 active projects and serve over 25,000 mobile users. The organization has a workforce of over 160 employees, including 40+ in research and development. In this case study, six developers contributed to the project, with their experience reflecting the average skill level within the company.

4.3.2. Execution of the Case Study

The case study began with a meeting between the measurers and the organization to understand their development process, which uses Scrum. In the organization, business analysts define requirements linked to Epic items, which are later broken down into Product Backlog Items (PBIs) and assigned to developers. We performed our measurements on 16 Epics. These selected epics contain a minimum of 1, a maximum of 6, and an average of 2.56 PBIs.

Two independent measurers measured the project's functional size using COSMIC and Event Points methods to ensure objectivity. After measuring the items using both methods, meetings were held to review assumptions, adjust measurements, and exclude any change requests. During the effort estimation process, two Epics were identified as possible outliers. After discussions with the organization, these outliers were removed, and the models were updated. The following section summarizes the result from the updated model constructed with 14 Epics.

4.3.3. Results

We performed linear regression analysis based on size measurement results. We used "total Event Point/COSMIC size" as the independent variable and "effort" as the dependent variable. Figure 4.7 shows the linear regression-based effort estimation models, including assessed residual plots, regression equations, and R-squared values. The analysis results show that Event Point demonstrates a strong relationship, and COSMIC demonstrates an adequate relationship with actual effort, considering the R-squared values.[99]

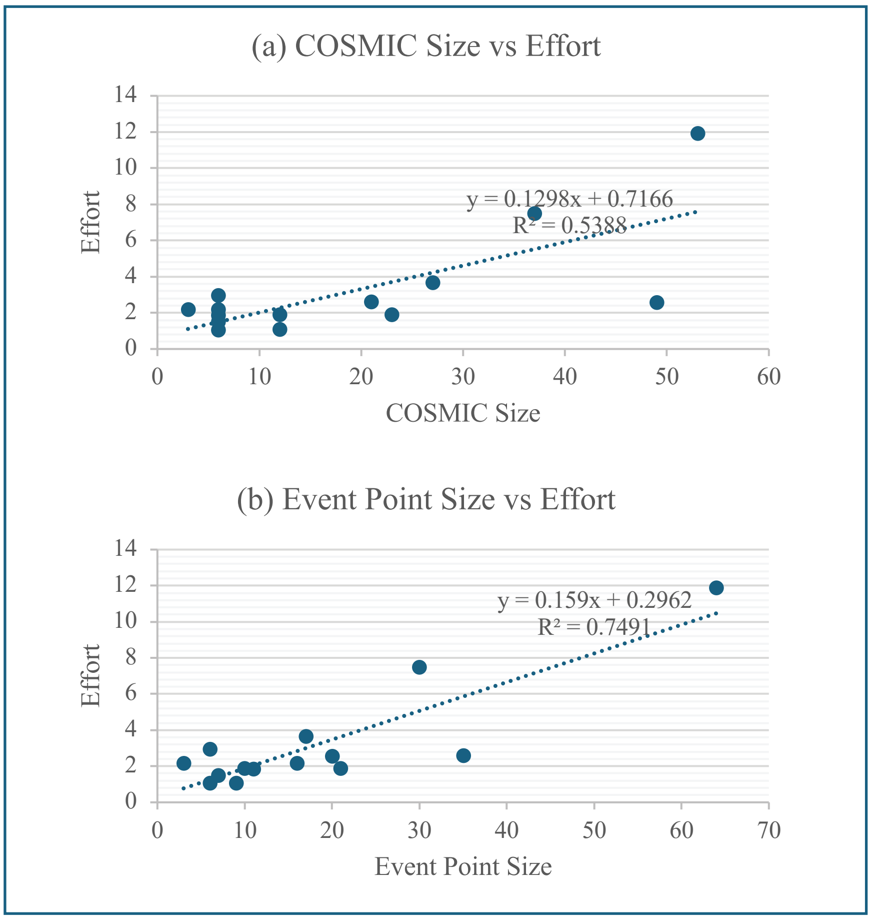

Figure 4.7. Linear regression-based effort estimation models for exploratory case study 1.

The evaluation results show that Event Point provided acceptable effort estimation, considering the MMRE metric.[101] On the other hand, COSMIC could not give an acceptable estimate of effort. Although the PRED(30) value for the Event Point model is higher than those for the COSMIC Size model, neither model met the threshold for successful estimation. A detailed summary of these values is presented in Table 4.3.

Table 4.3. Exploratory case study 1 results.

Method	**Regression**	**MMRE**	**MdMRE**	**PRED(30)**
COSMIC	Linear Regression	0.51	0.35	0.31
Event Point	Linear Regression	0.41	0.32	0.43

4.3.4. Implications from the Exploratory Case Study 1

This case study demonstrated that event-based size measurement provides better effort estimation results in MS-based projects than data movement-based measurement methods. This indicates the potential to develop an event-based size measurement approach tailored explicitly for microservice architectures.

Upon examining the evaluation metrics, although the Event Point method outperforms the COSMIC method, it does not demonstrate the ability to produce reliable predictive effort estimations. While an MMRE value of 0.41 falls within the acceptable range, it does not meet the rigorous standards required for a practical effort estimation model in the software industry. Thus, despite its relative improvement, the model lacks the precision necessary for widespread adaptation in predictive scenarios. There are two primary reasons behind this: (1) the actual effort provided by the organization was derived through several calculations rather than being directly tracked, and (2) the requirements used for measurement contained excessive technical details.

One factor contributing to the lack of precision in effort estimation is the organization's use of sprint-based effort tracking. Since the measurements were done at the PBI level, actual effort data for each PBI was needed. However, because such data was not consistently available, effort for each PBI had to be calculated using sprint-based actual effort and estimated PBI level effort provided by the organization. This calculated effort was then used as input for the model. For more accurate effort estimation models, it's crucial for organizations using agile methods to record actual effort at the PBI level or smaller work units. We recommended that the organization track effort data at the lowest level to improve accuracy in future estimations.

Another factor for the less-than-expected accuracy in effort estimation was the highly technical nature of the PBIs being measured. In FSM methods, requirements are expected to be defined at a functional level. However, due to the nature of Agile, requirements are broken down into PBIs, which often include design-phase decisions. As a result, while measuring these PBIs, we encountered both functional requirements and technical details, such as communication between microservices and algorithmic specifics. This led to certain abstractions in both methods during measurement, which may have impacted the reliability of the results.

Consequently, we observed that in Agile organizations working with MS-based projects, a measurement method capable of accurately assessing projects throughout all stages of the SDLC should also include size units at the design level. The following section describes the initial version of the MicroM method, which was developed based on the implications derived from this case study.

4.4. MicroM – The Initial Version

The MicroM method expresses software size in terms of events, which are the base components of microservices. While developing the method, we aimed to answer the question: "*What types of events can be used as base components at different levels of software representation?*"

MicroM categorizes the concept of events by considering their MS-based characteristics, as well as the limitations of FSM methods. MicroM defined three event categories: Interaction, Communication, and Process (see Figure 4.8).

Interaction Events encompass events triggered by user actions within the flow and any resulting outputs from the system to the user, primarily focusing on front-end layer activities. These events can be mapped to the COSMIC method's Entry and Exit data movements, but without relying on the Object of Interest (OOI) concept. Instead, interactions are measured by counting events.

Communication Events involve events that arise from publishing, subscribing, and messaging activities, mainly covering backend communication between microservices. The communication event is designed as a specific category tailored to MSSAs.

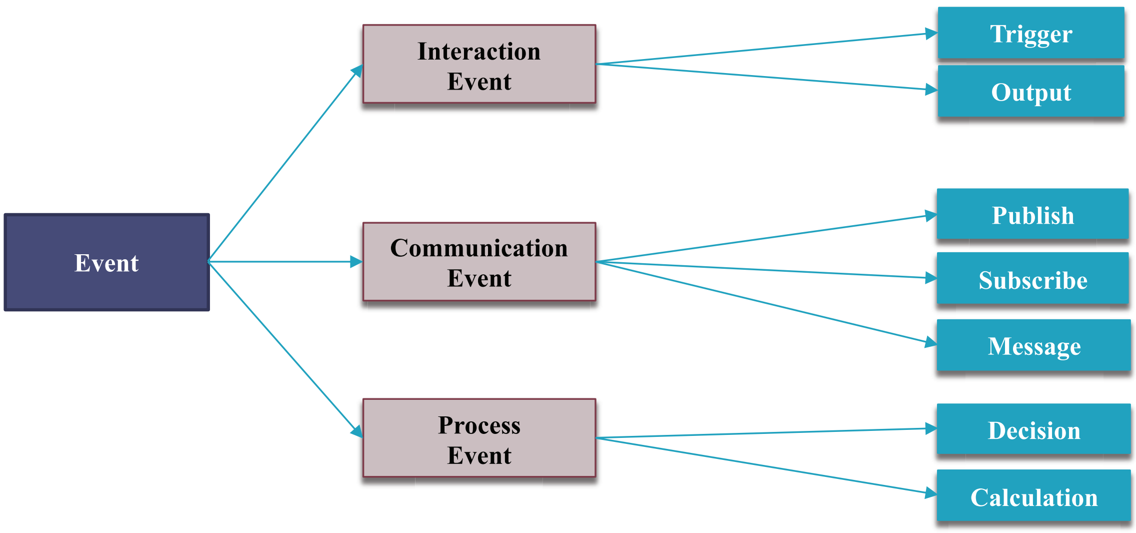

Figure 4.8. MicroM size measurement method (version 1).

Process Events encompass the calculations and decision-making events that occur during the flow, primarily focusing on backend layer activities. The calculation activities include tasks related to data manipulation, filtering, reporting, and file operations. Decision-making activities cover decision structures such as "OR", "XOR", and "AND", This category introduced by MicroM addresses gaps present in FSM methods like COSMIC, offering significant value in software development, particularly in capturing and representing algorithmic complexities.

The MicroM method recommends using eEPC diagrams to define events (see Section 2.1.3). Different abstraction levels of the software representations can be employed during each measurement process, considering the project characteristics; however, this abstraction must be maintained at a consistent level throughout the entire measurement process. Therefore, the first step of the measurement should involve determining the appropriate level of abstraction. In order to identify communication events in MicroM size measurement, the system must be decomposed into microservices. Additionally, process events can only be determined if the requirements are defined at a detailed level. Therefore, efficient MicroM measurement requires that design-level decisions have been made.

The process of measuring the size of a software project begins with a thorough analysis of the project's requirements, followed by identifying the specific requirements to be measured. Once identified, these requirements are conceptually categorized into three distinct types of events: Interaction, Communication, and Process Events. Each of these events is then counted separately based on their respective categories. The overall size of the requirement is determined by aggregating the total number of identified events; alternatively, it can be expressed as totals for each category. This approach offers a more nuanced understanding of the software's complexity and functionality. The following section presents the exploratory case study conducted to evaluate the success of the initial MicroM method.

4.5. Exploratory Case Study 2

The details of Exploratory Case Study 2, conducted to assess the success of the MicroM method we developed and to examine the benefits of the recommendations we made in the project from Exploratory Case Study 1, are provided in the sections below.[111]

4.5.1. Case Study Design

This case study aims to assess the success of our proposed MicroM method, which categorizes the event in a more microservice-centric approach. We also sought to evaluate how our previous recommendations improved the size measurement and effort estimation processes within the agile organization involved in Exploratory Case Study 1. To achieve this, we formulated the following research questions:

- *RQ1. How successful/precise is the proposed event categorization in the size measurement and effort estimation of MS-based projects?*
- *RQ2. What other purposes is this categorization useful for?*

4.5.1.1. Case Selection Criteria

The most critical case selection criterion in this study is the selection of requirements for which actual effort was entered, and descriptions were written as a result of the recommendations we made to the organization in Exploratory Case Study 1.

4.5.1.2. Description of the Case

The project involved in this case is the same as the one in Exploratory Case Study 1. However, the effort was recorded on a PBI basis in this case, and regular descriptions were written for the PBIs as part of our recommendations.

4.5.2. Execution of the Case Study

The case study began with a meeting between the measurers and the organization to determine the requirements to be measured, during which it was revealed that the organization had started recording actual efforts for PBIs, as recommended in Exploratory Case Study 1. However, since the project was completed during this process, most PBIs for which effort was recorded were change requests. In this meeting, 17 non-outlier PBIs were selected for size measurement.

The project's functional size was measured using COSMIC and MicroM methods by three independent and experienced measurers who were not involved in the development process, ensuring objectivity. To maintain consistency, they jointly measured the size of 17 PBIs and discussed assumptions before finalizing the measurements. Data collected during the case study included COSMIC size, MicroM size (categorized by Process, Communication, and Interaction), PBI effort in person-hours, and any encountered difficulties. The process was followed by the creation of an effort estimation models. The following section summarizes the result from the effort estimation models constructed with 17 PBIs.

4.5.3. Results

We first performed linear regression analysis based on size measurement results. We used "total MicroM/COSMIC size" as the independent variable and "effort" as the dependent variable. Figure 4.9 shows the linear regression-based effort estimation models, including assessed residual plots, regression equations, and R-squared values. The analysis results show that Event Point demonstrates a strong relationship, and COSMIC demonstrates an adequate relationship with actual effort, considering the R-squared values.[99]

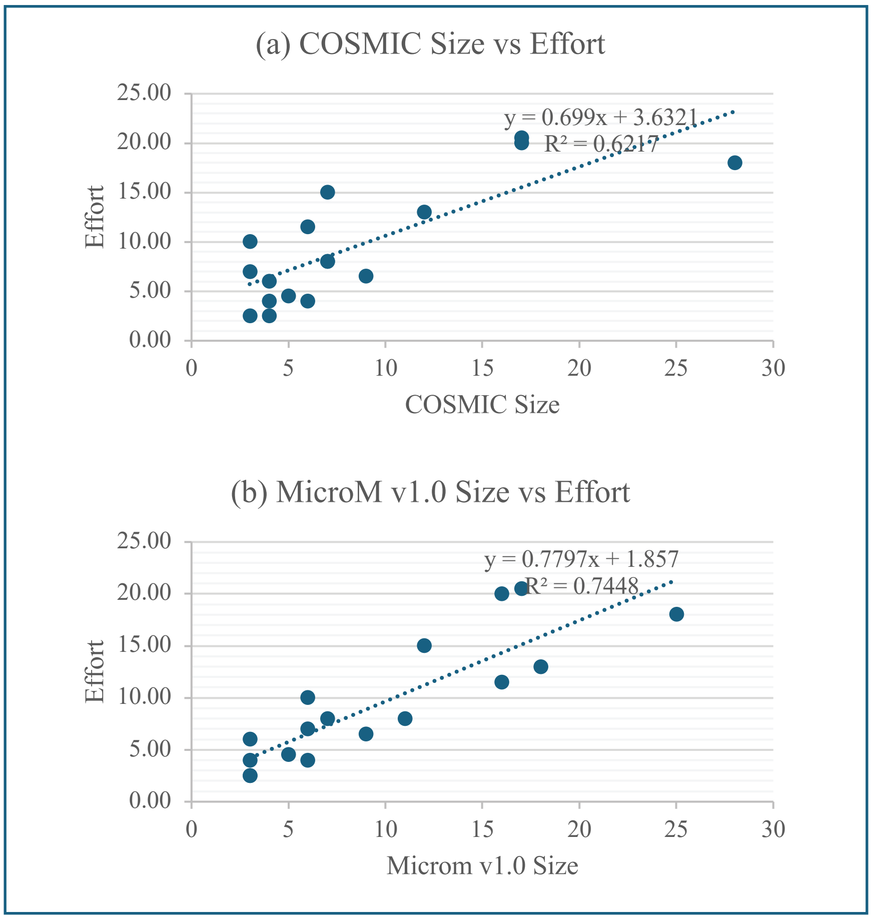

Figure 4.9. Linear regression-based effort estimation models for exploratory case study 2.

We performed multiple linear regression to evaluate the impact of each COSMIC data movement and MicroM event category on effort. We used three "MicroM event categories (Interaction, Communication, and Process) and four "COSMIC data movement types (Entry, Exit, Read, and Write)" as the independent variables and "effort" as the dependent variable. We used Pearson's correlation coefficient to measure correlation. The multiple regression analysis results, including intercept, slope coefficients, R-squared, and Significance F values of the projects, are presented in Table 4.4. The analysis results show that considering the R-squared values, the MicroM method demonstrates a higher correlation with actual effort than the COSMIC method.

Table 4.4. Multiple linear regression analysis for exploratory case study 2.

	COSMIC	**MicroM v1.0**
Number of Observations	17	17
Intercept	2.72	1.72
Slope Coefficient for Entry	0.25	-
Slope Coefficient for Read	-0.31	-
Slope Coefficient for Write	0.70	-
Slope Coefficient for Exit	2.12	-
Slope Coefficient for Interaction	-	1.06
Slope Coefficient for Communication	-	0.22
Slope Coefficient for Process	-	0.94
R-squared	0.69	0.76
Significance F	0.00	0.00

The evaluation results show that all models provided acceptable[101] effort predictions (see Table 4.5). The MicroM models performed better than the COSMIC models, approaching the level of predictive accuracy with multiple linear regression. We observed a 13% improvement in the MMRE (Mean Magnitude of Relative Error) value in the MicroM-based model created with multiple linear regression compared to linear regression. We also obtained better results with the MicroM method than the COSMIC method based on the MdMRE and PRED(30) metrics.

Table 4.5. Exploratory case study 2 results.

Method	**Regression**	**MMRE**	**MdMRE**	**PRED(30)**
COSMIC	Linear Regression	0.46	0.32	0.47
	Multiple Linear Regression	0.47	0.27	0.53
MicroM v1.0	Linear Regression	0.31	0.28	0.71
	Multiple Linear Regression	0.27	0.26	0.59

4.5.4. Implications from the Exploratory Case Study 2

In this study, we aimed to achieve two primary objectives: first, to evaluate the success of our proposed MicroM size measurement method for MS-based projects, and second, to explore additional applications of this categorization. Our previous research (Exploratory Case Study 1) showed that event-based size measurement yielded more accurate effort estimations than COSMIC, highlighting the importance of aligning measurement methods with the assessed architecture.

We developed a MicroM size measurement method tailored for microservices, categorizing events according to the architecture. This method enabled us to create a more accurate effort estimation model than the COSMIC-based one. However, although we achieved more accurate results compared to the COSMIC method, we were unable to create a predictive effort estimation model (MMRE=0.27 for MicroM). This could likely be attributed to the nature of the measured PBIs, most of which were change requests.

Despite this, the MicroM measurement approach offers insights beyond effort estimation, particularly regarding coupling complexity within projects. By analyzing the ratio of communication events to total events, we can gauge coupling complexity; an increase in this ratio signifies heightened coupling, which can complicate management among microservices and make implementing change requests more challenging. This insight can aid in decision-making regarding change request approvals by considering both effort and complexity.

Additionally, we observed improvements in the organization's practices. Our previous study identified specific challenges in size measurement and effort estimation in agile environments, such as the lack of recorded actual effort at the PBI level and insufficient descriptions. Our recommendations to include detailed descriptions and record actual effort at the PBI level proved effective in this study, leading to quicker and more precise measurements and effort estimations based on actual data.

In this study, we also evaluated the suitability of the developed method for MSSAs. We found that the event categories we defined exhibited varying correlations with effort. The multiple regression model we developed yielded more accurate effort estimations than the linear regression model based on the total number of events. However, we recognized that each microservice might have its own database, and we had not accounted for database interactions in our event categories. While the initial version of our method is suitable for measurement during the design phase of the SDLC because it incorporates design-level decisions, it should ultimately be applicable across all SDLC phases, including the requirements phase, to enhance its utility.

4.6. Summary of the Chapter

In the literature review, we observed a transition in today's software world from monolithic architectures, database, and transaction-oriented structures to MS-based, more behavioral, and event-oriented architectures. However, the objective FSM methods, which have been successfully used in traditional monolithic architectures, may not be able to respond to measuring the behavioral and event-oriented structure in MSSAs, as they typically rely on a data movement-based counting approach. In the surveys and interviews we conducted after the literature review, we found that organizations face difficulties when performing measurements in MS-based projects and are unable to use objective measurement methods.[37,38] These findings have led to the development of an objective sizing measurement method for MSSAs.

In the method we developed, we drew inspiration from events, which are the base components of microservices. Accordingly, we first conducted an exploratory case study to evaluate the success and suitability of the Event Point[51] method in MSSAs. As a result of this study, we found that the event-based measurement method performed better in MS-based projects compared to the data movement-counting FSM method[33]. However, when counting events in the method to be developed, it is essential to consider the characteristics of MSSAs.

In the first version of the MicroM, considering the shortcomings of existing sizing measurement methods, we categorized events into three groups: interaction, communication, and process events. In this initial version, interaction events cover user interactions, communication events cover communication between the microservices, and process events cover calculation and decision. We evaluated this version by conducting a second exploratory case study on an MS-based project. Similar to the first exploratory case study, it yielded better results than the COSMIC method. Additionally, we observed that the event categories we defined had different correlations with effort. The multiple regression model we created provided a more accurate effort estimation than the linear regression model that used the total number of events. However, during this study, we realized that each microservice could have its own database, and we had not defined an event category that accounts for database interactions. The initial version of the method is suitable for measurement during the SDLC design phase as it includes design-level decisions. However, for the method to be useful, it should be measurable across all phases of the SDLC, such as requirements level. We improved the first version method and created MicroM version 2, which is explained in the next chapter.

CHAPTER 5

MICROM: A SIZE MEASUREMENT METHOD FOR MICROSERVICE-BASED ARCHITECTURES

In this chapter, we introduce the most recent version of MicroM. We developed the final version based on the findings from the performed phases of our research methodology. These findings are as follows:

- In the software industry, there is a paradigm shift moving from the database and transaction-oriented structures, as seen in "monoliths," to more behavioral and event-oriented structures, as in "microservices".
- Objective FSM methods, effective in traditional monolithic architectures, may struggle with measuring behavioral and event-oriented structures in MSSAs due to their reliance on data movement-based counting approaches.
- Organizations face challenges in applying objective size measurement in MS-based projects. There is no de facto standard method used in the industry for microservice projects.
- There is a correlation between size based on events, the base components of MS-based projects, and effort.
- The event categories (interaction, communication, and process) defined in MicroM version 1 have different correlations with effort.
- MicroM version 1 does not include events that count interactions with databases; it only counts communication between microservices. In an MSSA, database operations should also be considered.
- MicroM version 1 requires the project to make design-level decisions to count the defined event categories. However, the developed method needs to be applicable throughout all phases of the SDLC.

Considering our findings, we aimed to answer the question: *"What types of events can be used as base components at different levels of software representation?"* The following section presents the improved version of the MicroM Size Measurement Method.

5.1. Description of MicroM Size Measurement Method

MicroM is an objective size measurement method tailored for MSSAs. It focuses on the events that form the foundation of microservices, categorizing these events into different abstraction levels of the software representations based on the architectural characteristics. This method is designed to be applicable throughout all stages of the SDLC.

MicroM classifies events into three abstraction levels: (1) Functional Level, (2) Architectural Level, and (3) Algorithmic Level (see Figure 5.1). The varying levels of abstraction in the method allow the MicroM approach to be utilized from the requirements phase through to the post-project stage of the SDLC. As the project details mature, the quality of the measurements improves.

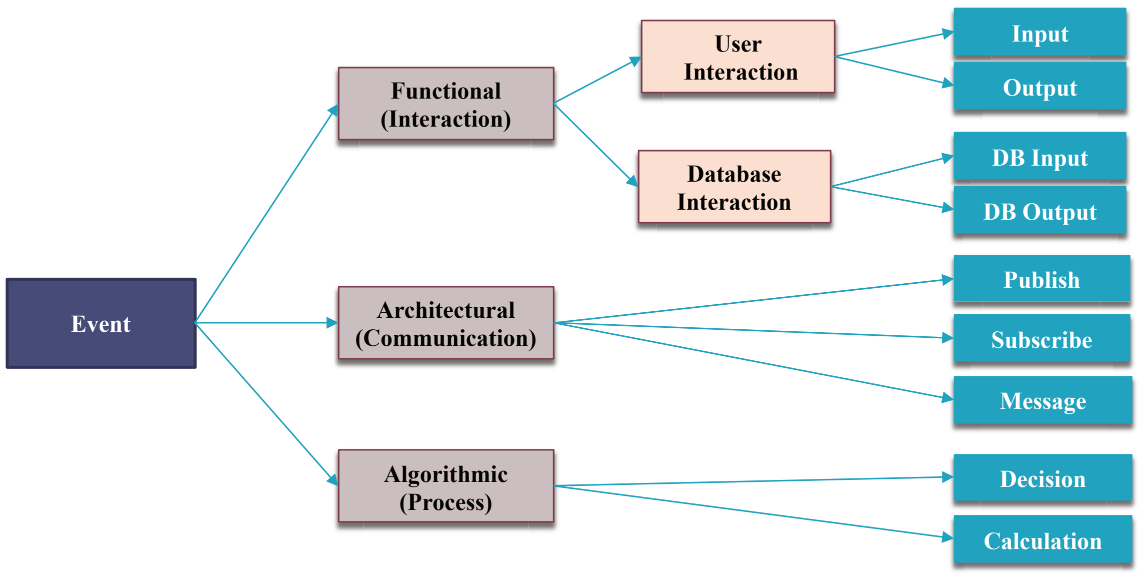

Figure 5.1. MicroM size measurement method.

The functional level covers the events during user and database interaction. User interaction events consist of user inputs and outputs while database interaction events consist of database inputs and database outputs. This level abstracts the system between the functional user and persistent storage. The concept of a functional user is as in the COSMIC method and refers to "a type of user that is a sender and/or an intended recipient of any data in the functional user requirements of a piece of software". Functional-level events can be mapped to the Entry, Read, Write, and Exit data movements in the COSMIC method. However, MicroM does not apply the OOI concept; instead, it measures interactions by counting events. In this method, interaction events include all the data that can be sent to the system by the user on a single screen or displayed by the system to the user on a single screen. For instance, filling in all fields on a form is counted as a single interaction event. Similarly, a form displayed to the user is also counted as a single interaction event, even if the data originates from different OOIs. Database interaction events are also at the same level of abstraction as user interaction events. For example, writing all the data entered through a single screen to the database or reading all the data that can be displayed to the user on a single screen from the database is counted as a single event. Functional-level event examples can be found in Figure 5.2.

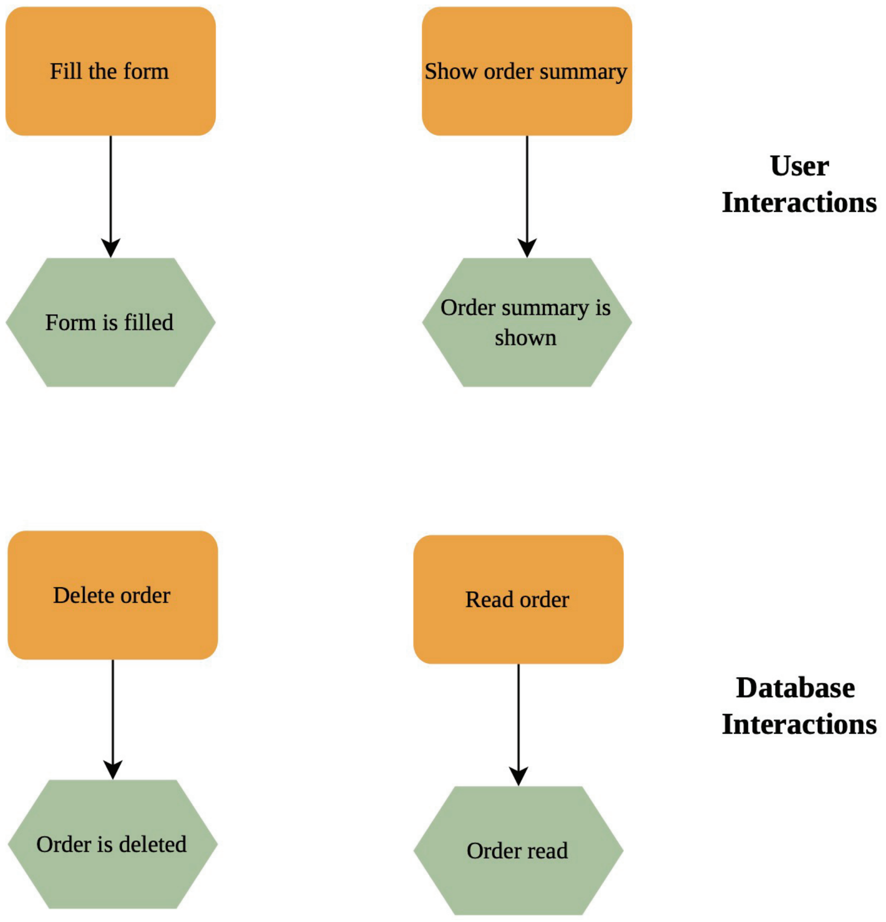

Figure 5.2. Functional level events.

The architectural level covers the events during communication within the system. Therefore, design-level decisions should already be made to identify events at this level. This abstraction level has been developed in the MicroM method as a specific level tailored for MSSAs. Architectural-level events consist of communication between microservices, including message exchanges, as well as events triggered by publish and subscribe mechanisms in event-based systems. At this level, in request-driven MSSA, each request-response pair is counted as one communication event. In event-driven MSSA, each message published to the event queue by a microservice and each subscription of a microservice to the event queue are also counted as one communication event. Architectural level event examples can be found in Figure 5.3.

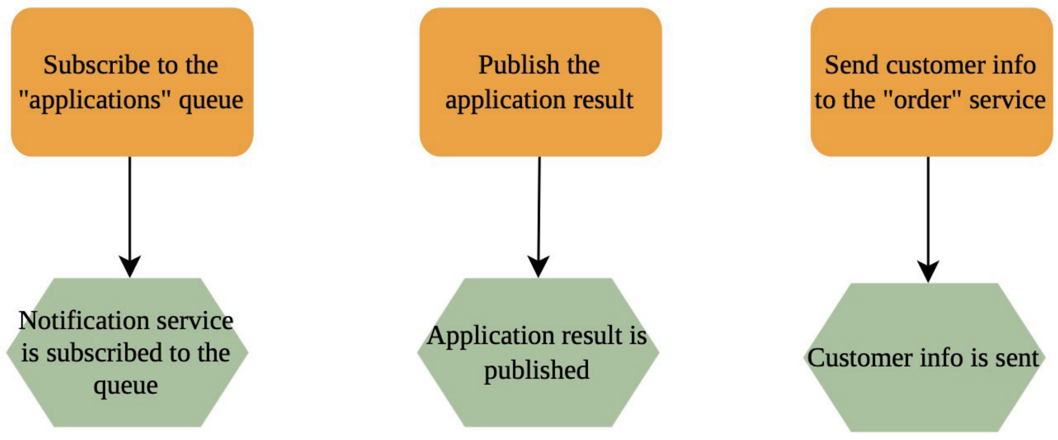
Subscribe to the "applications" queue
Publish the application result
Send customer info to the "order" service
Notification service is subscribed to the queue
Application result is published
Customer info is sent

Figure 5.3. Architectural level events.

The algorithmic level covers the events during the processing issues, such as decision and calculation. This level introduced by MicroM addresses aspects missing in FSM methods like COSMIC but holds significant value in software development, particularly in representing algorithmic complexities. Decision events include events that result from decision mechanisms such as "OR", "XOR", and "AND", while calculation events encompass tasks involving data manipulation, such as arithmetic calculations, filtering, reporting, and file creation. Events at this level are algorithmic events that occur at the functional requirement level. For example, each event resulting from a decision determined at the functional requirement level is counted as a decision event (e.g., yes/no, younger than 18 years/older than 18 years, etc.). Unlike the COSMIC method, events resulting from tasks such as report generation in different templates, creating files in different formats, and data manipulation involving calculations like tax, discount, etc., are counted as one calculation event. Algorithmic level event examples can be found in Figure 5.4.

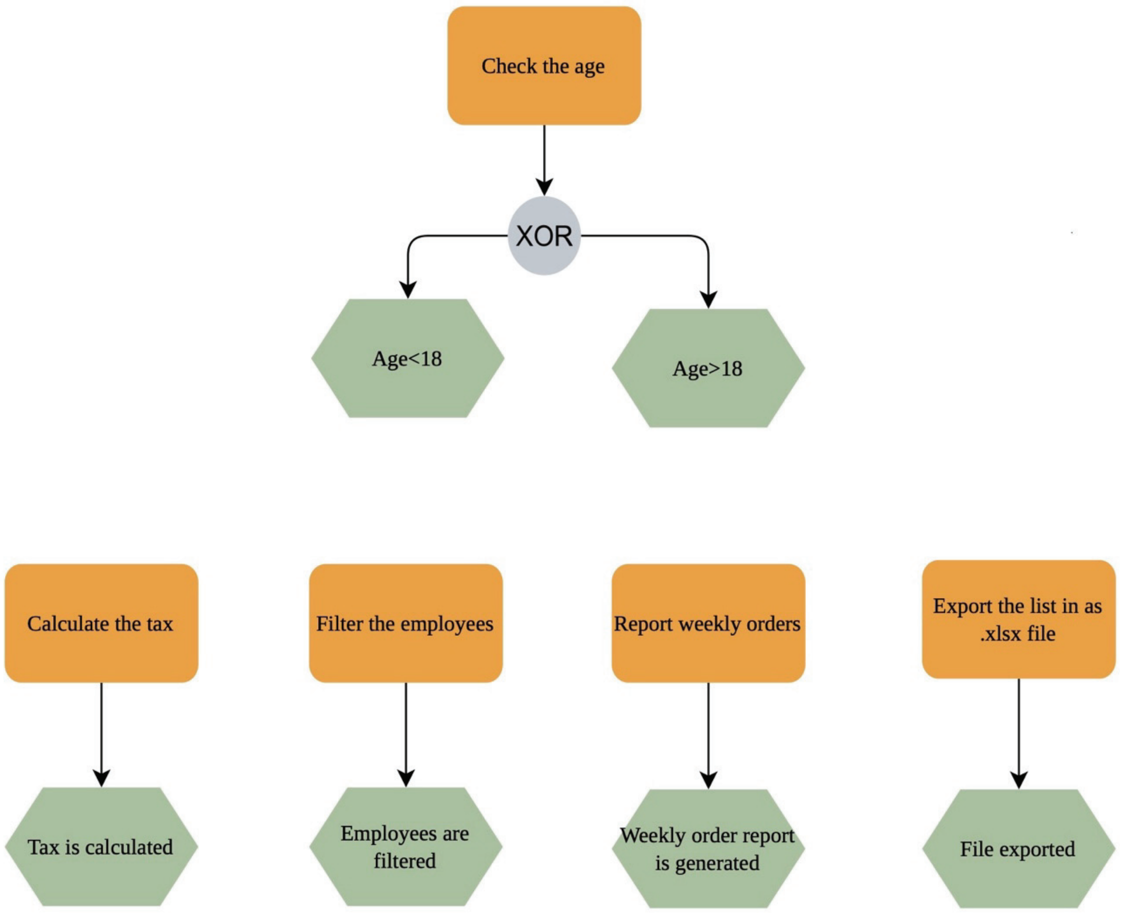

Figure 5.4. Algorithmic level events.

In summary, MicroM's classification of events into Functional, Architectural, and Algorithmic Levels provides a clear framework for measuring the software size of MS-based systems throughout the SDLC. Each level focuses on different aspects: the Functional Level deals with user and database interactions, the Architectural Level addresses communication between microservices, and the Algorithmic Level covers decision-making and data-processing tasks. These levels help to provide a more detailed view of software size, complementing the existing FSM methods and offering a valuable approach for sizing MSSAs.

5.2. Application of MicroM Size Measurement Method

The application of the MicroM size measurement method to project requirements involves the following steps (see Figure 5.5):

A. The software project's requirements are analyzed, and the requirements that will be measured are determined. The abstraction level of the requirements is defined. Based on the defined abstraction level, the MicroM levels to be included in the measurement scope are determined. For example, since determining architectural level events at the requirements stage can be challenging, they may be excluded from the scope of measurement.
B. Each identified requirement is modeled using the eEPC notation. Events in the model are conceptually categorized into Interaction, Communication, and Process Events according to the MicroM rules.
C. The identified events are counted separately based on their levels.
D. The size of the requirement is determined by the total number of events.
E. Alternatively, the size of the requirement is expressed as totals for different levels.

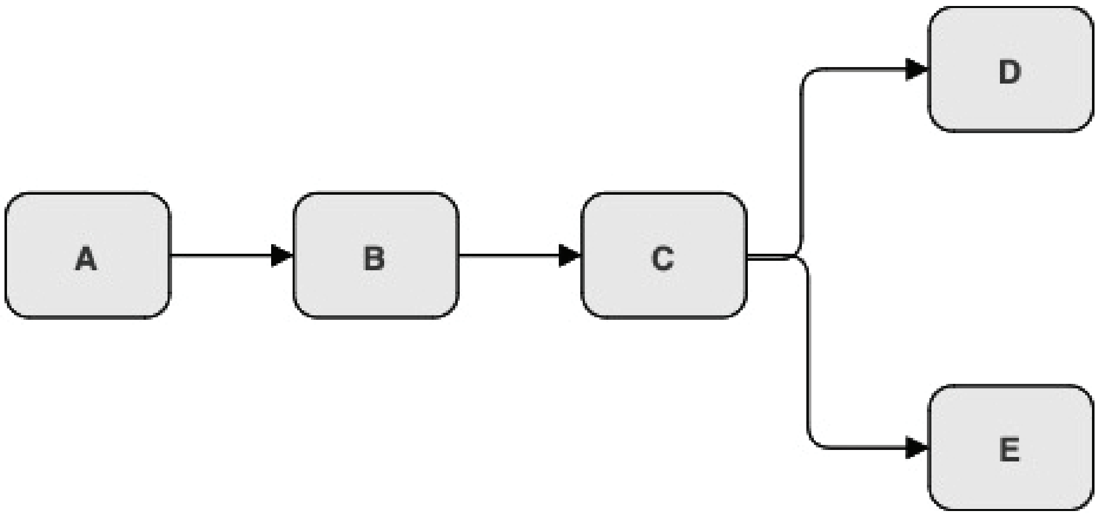

Figure 5.5. Application of MicroM.

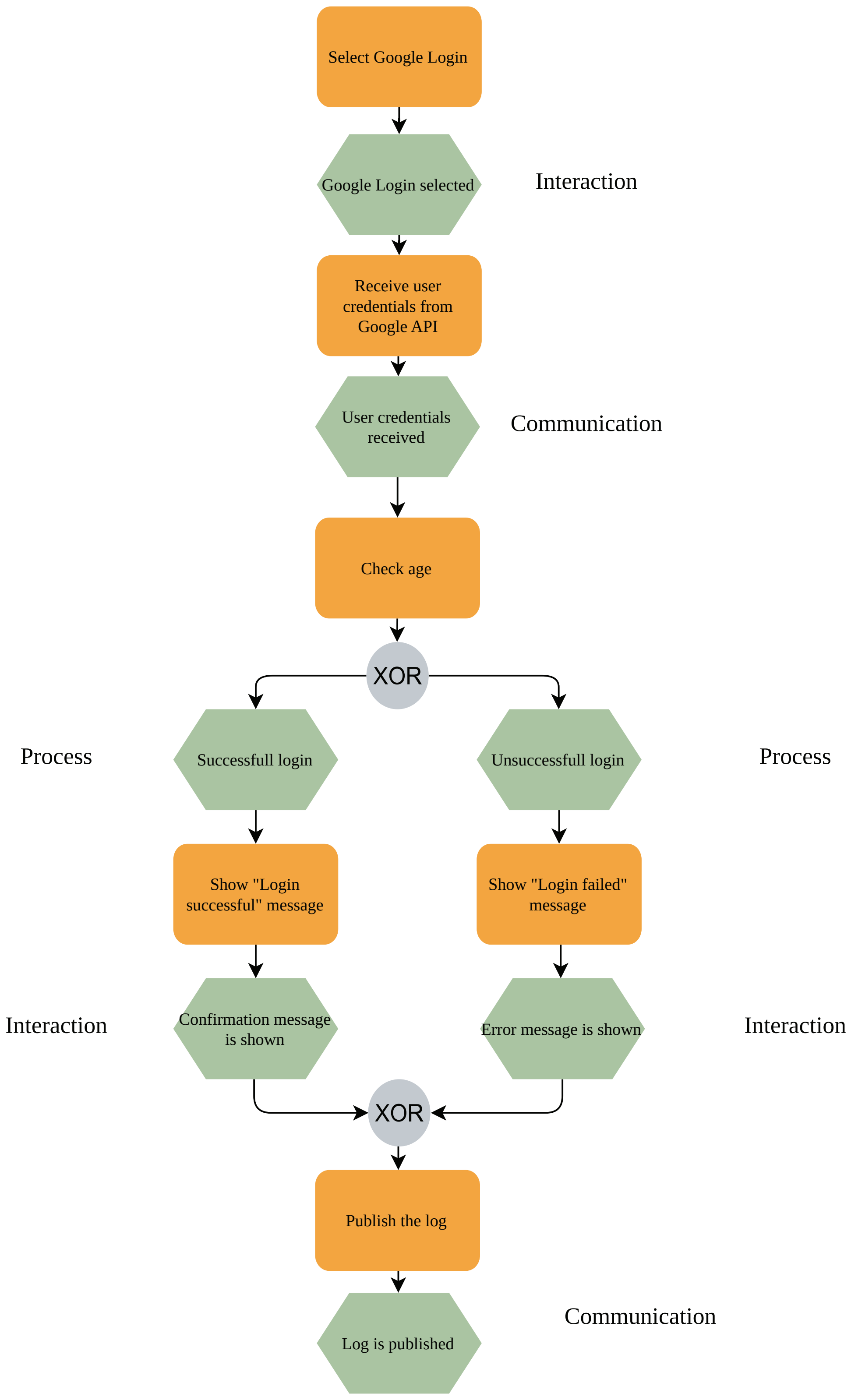

Figure 5.6. Application of MicroM – example 1.

These application steps are illustrated with the following two examples. The first example is a login process, and the requirement is identified as "*When the user enters their email address and password to log into the system, the service checks the credentials from the database. If the credentials are correct, a "Login successful" message is displayed to the user, and if incorrect, a "Login failed" message is shown, and the status is published to log*.". The events for the requirement are identified in Figure 5.6. The MicroM size for this requirement is determined to be a total of 7 events: 4 Interaction events, 2 Process events, and 1 Communication event.

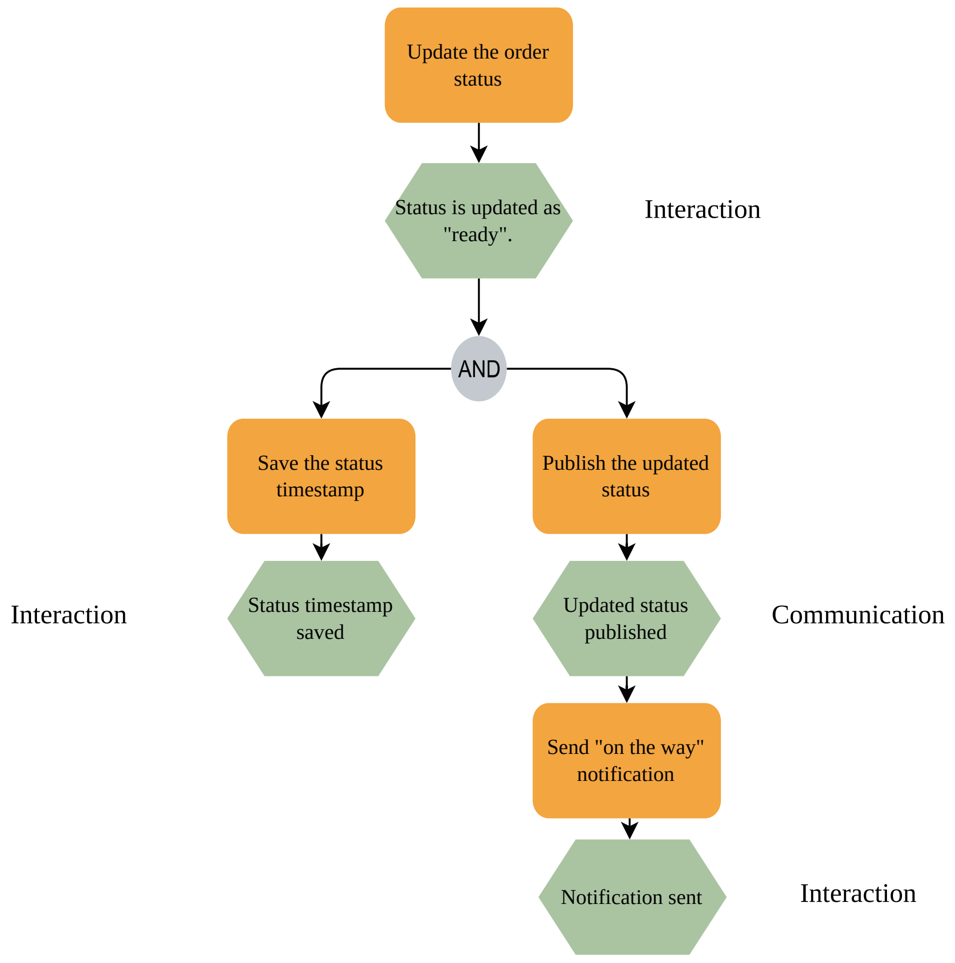

Figure 5.7. Application of MicroM – example 2.

The second example is from a food order application. The selected requirement is about a notification process. The requirement is identified as "When the order is ready, the restaurant changes the status of the order. The order service writes the timestamp to the database and publishes the new status. The notification service sends the customer a "your order is on the way" push notification.". The events for the requirement are identified in Figure 5.7. The MicroM size for this requirement is determined to be a total of 4 events: 3 Interaction events, and 1 Communication event.

5.3. Summary of the Chapter

In this chapter, we introduced the final version of the MicroM method, a size measurement approach designed explicitly for MSSAs. It addresses the limitations of traditional FSM methods by focusing on events as the base components of microservices, categorizing them into three abstraction levels of the software representation: Functional, Architectural, and Algorithmic. These levels encompass user interactions, communication between microservices, and processing tasks. MicroM is applicable throughout the entire SDLC with the quality of measurements improving as project details mature. We also provided examples from a login process and a food order notification to demonstrate the method in practice. We showed how it can accurately measure software size based on event counting, offering a tailored solution for MS-based systems. The next chapter provides a detailed analysis of the results from evaluating the final version of the model.

CHAPTER 6

EVALUATION

In this chapter, we first provide a detailed analysis of the results from evaluation case studies. Then, we discuss our findings and mention the possible threats to validity.

6.1. Evaluation Case Studies

In the following sections, we describe the details of the case studies conducted to evaluate the MicroM Size Measurement Method.

6.1.1. Case Study Design

These case studies aim to assess the success and reliability of the size measurement method proposed in the "solution design" phase. To guide this evaluation, we aimed to determine the effectiveness of the proposed MicroM size measurement method for estimating effort in MS-based projects. For this aim, we defined the following research questions:

- *RQ1. Is MicroM relevant as a size measure for MSSAs?*
- *RQ2. Is MicroM relevant as a size measure for traditional monolith architectures?*
- *RQ3. How does MicroM perform in estimating effort compared to other size measurement methods?*

6.1.1.1. Case Selection Criteria

We selected our cases based on specific criteria. The first criterion includes projects developed with MSSAs and a counter case that features a project developed using traditional monolithic architecture. The second criterion is that these projects should have software artifacts with enough details to be measurable and related effort data. The maturity of the documentation is essential for reliable evaluation, and access to relevant organizations is necessary if needed. The last criterion is to include organizations that develop projects using both Agile and Waterfall methodologies to evaluate the success and adaptability of the MicroM method across different SDLC approaches.

6.1.1.2. Description of the Cases

We conducted three evaluation case studies to answer the research questions. Based on the defined criteria, Cases 1 and 2 involve projects developed with MSSAs. In contrast, Case 3 is the counter case, involving a project developed with a traditional monolithic architecture. This allowed us to assess how MicroM performs in non-microservice architectures. In Cases 1 and 2, the organizations use Agile methodology, while in Case 3, the Waterfall methodology is used.

In all cases, we evaluated the success of effort estimation models using both MicroM and COSMIC methods. Table 6.1 summarizes the cases, and the following sections describe each case study in detail.

Table 6.1. Summary of the evaluation case studies.

	Case 1	**Case 2**	**Case 3**
Domain	Finance	MIS	Aviation
Architecture	Microservice	Microservice	Monolith
Development Type	Backend	Full Stack	Full Stack
Documentation Level	Design	Design	Requirement
Measured Artifacts	Issue (Jira)	Task (Notion)	Requirement Document
Measurement Methods	MicroM & COSMIC	MicroM & COSMIC	MicroM & COSMIC
Number of Samples	45	17	14

6.1.2. Evaluation Case Study 1

Organization 1 is a software development and consultancy company with over 25 years of experience. The project included in the case study is in the finance sector and is being developed using an event-based MSSA. The organization adopts Agile methodology and is primarily responsible for the backend development of the project. The project is currently in development, and the measurements were conducted based on completed issues in Jira. The actual efforts are recorded in Jira on an issue-by-issue basis as person-hours.

6.1.2.1. Execution of the Case Study

The case study began with a meeting between the measurer and the organization to identify the issues to be measured. In this meeting, 45 issues were selected, all of which have actual effort data, are completed, involve new development tasks, and have enough descriptions for measurement. In the first cycle, we measured the issues and constructed effort estimation models for both MicroM and COSMIC. In a later meeting with the organization, the assumptions made during the measurement process were clarified, during which we identified an outlier issue considering the productivity. This outlier issue was completed in more time than expected, and it was noted that the reason was the complexity of the task, which was being done for the first time. Consequently, new effort estimation models were developed using the remaining 44 issues. The following section summarizes the result from the updated model constructed with 44 issues.

6.1.2.2. Results

We first performed linear regression analysis based on size measurement results. We used "total MicroM/COSMIC size" as the independent variable and "effort" as the dependent variable. Figure 6.1 shows the linear regression-based effort estimation models, including assessed residual plots, regression equations, and R-squared values.

The analysis results show that MicroM demonstrates an acceptable relationship with actual effort, considering the R-squared values.[99] On the other hand, the relation between COSMIC size and effort is not reliable for planning.

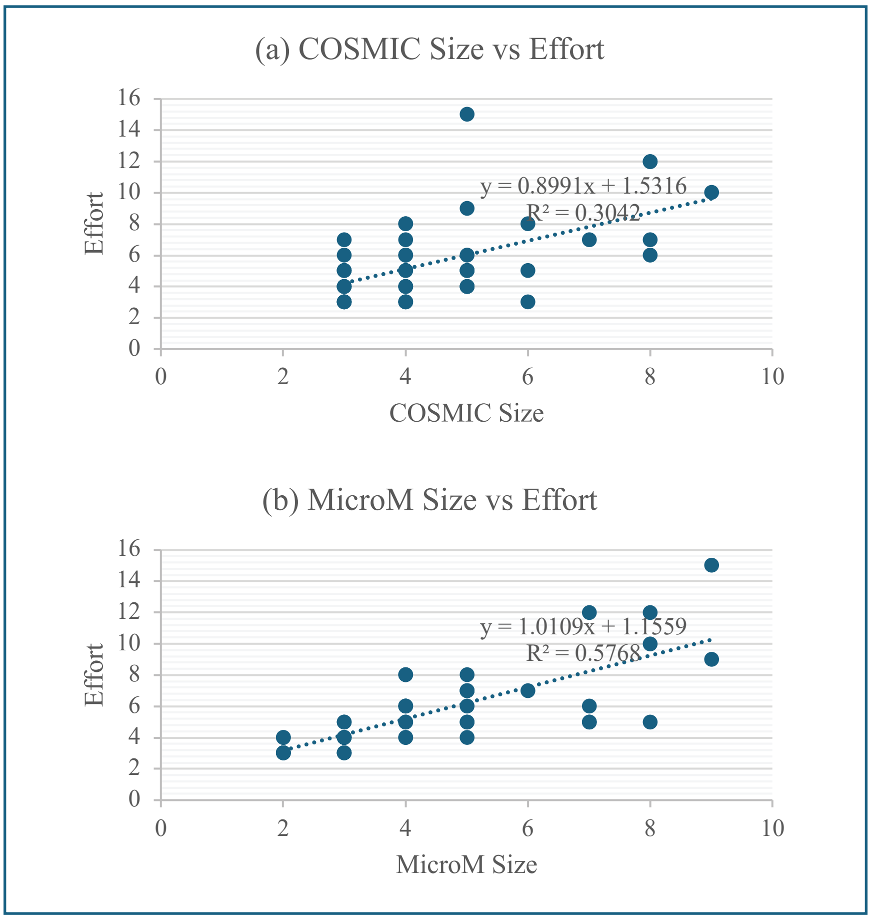

Figure 6.1. Linear regression-based effort estimation models for case study 1.

We performed multiple linear regression to evaluate the impact of each COSMIC data movement and MicroM event level on effort. We used three "MicroM event levels (Interaction, Communication, and Process) and four "COSMIC data movement types (Entry, Exit, Read, and Write)" as the independent variables and "effort" as the dependent variable. We used Pearson's correlation coefficient to measure correlation. The multiple regression analysis results, including intercept, slope coefficients, R-squared, and Significance F values of the projects, are presented in Table 6.2.

Table 6.2. Multiple linear regression analysis for evaluation case study 1.

	COSMIC	MicroM
Number of Observations	44	44
Intercept	1.49	1.49
Slope Coefficient for Entry	0.52	-
Slope Coefficient for Read	1.17	-
Slope Coefficient for Write	0.80	-
Slope Coefficient for Exit	1.13	-
Slope Coefficient for Interaction	-	1.06
Slope Coefficient for Communication	-	0.74
Slope Coefficient for Process	-	1.22
R-squared	0.32	0.59
Significance F	0.00	0.00

The multiple linear regression-based analysis results show that the R-squared value was improved for both COSMIC and MicroM. The models constructed with the MicroM method represent adequate relationships considering the R-squared values.[99] On the other hand, although some improvements occurred, the models with the COSMIC method could not present a reliable relationship with the effort.

The evaluation results of the regression-based models are presented in Table 6.3. In this case, all models provided acceptable effort predictions, considering MMRE metric.[101] The MicroM models performed better accuracy than the COSMIC models, approaching the level of predictive accuracy (MMRE=0.21 for multiple linear regression). The results also indicate that multiple linear regression did not show improvement for the COSMIC method. However, for the MicroM method, while the improvement was not substantial, there was some noticeable enhancement in terms of MMRE and MdMRE.

Table 6.3. Evaluation case study 1 results.

Method	Regression	MMRE	MdMRE	PRED(30)
COSMIC	Linear Regression	0.31	0.28	0.59
	Multiple Linear Regression	0.31	0.28	0.52
MicroM	Linear Regression	0.22	0.18	0.73
	Multiple Linear Regression	0.21	0.13	0.70

6.1.3. Evaluation Case Study 2

Organization 2 is a software development company with over 30 years of experience. The project included in the case study is an MIS project and is being developed using an MSSA. The organization adopts the Agile methodology. The project is currently in development, and the measurements were conducted based on completed tasks in Notion. The actual efforts are recorded in Notion task-by-task as person-hours.

6.1.3.1. Execution of the Case Study

The case study began with a meeting between the measurer and the organization to identify the issues to be measured. In this meeting, 17 tasks were selected, all of which have actual effort data, are completed, involve new development tasks, and have enough descriptions for measurement. In the first cycle, we measured the tasks and constructed effort estimation models for both MicroM and COSMIC. In a later meeting with the organization, the assumptions made during the measurement process were clarified, during which we identified two outlier tasks considering productivity. These outlier tasks were completed in less time than expected, and it was noted that the reason was the repeated code. Consequently, new effort estimation models were developed using the remaining 15 tasks. The following section summarizes the result from the updated model constructed with 15 tasks.

6.1.3.2. Results

We first performed linear regression analysis based on size measurement results. We used "total MicroM/COSMIC size" as the independent variable and "effort" as the dependent variable. Figure 6.2 shows the linear regression-based effort estimation models, including assessed residual plots, regression equations, and R-squared values. The analysis results show that the relation between both methods and effort is not reliable for planning, considering the R-squared values.[99]

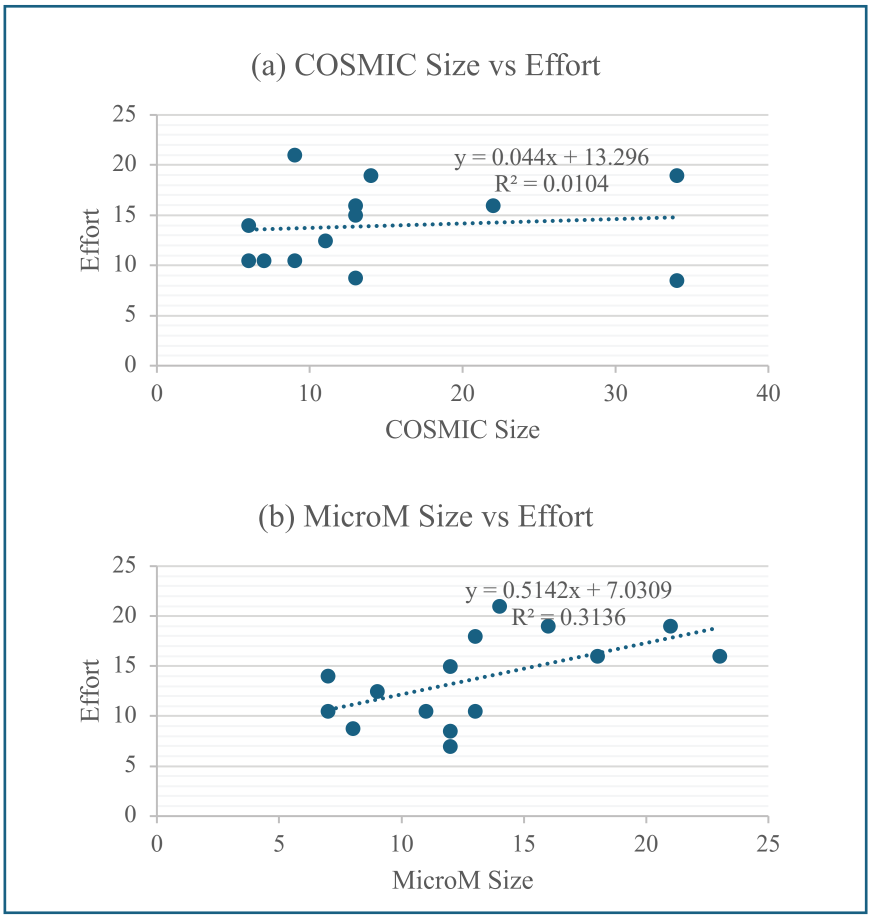

Figure 6.2. Linear regression-based effort estimation models for case study 2.

We performed multiple linear regression to evaluate the impact of each COSMIC data movement and MicroM event level on effort. We used three "MicroM event levels (Interaction, Communication, and Process) and four "COSMIC data movement types (Entry, Exit, Read, and Write)" as the independent variables and "effort" as the dependent variable. We used Pearson's correlation coefficient to measure correlation. The multiple regression analysis results, including intercept, slope coefficients, R-squared, and Significance F values of the projects, are presented in Table 6.4

Table 6.4. Multiple linear regression analysis for evaluation case study 2.

	COSMIC	MicroM
Number of Observations	15	15
Intercept	8.94	7.52
Slope Coefficient for Entry	1.26	-
Slope Coefficient for Read	1.33	-
Slope Coefficient for Write	0.19	-
Slope Coefficient for Exit	-1.13	-
Slope Coefficient for Interaction	-	0.21
Slope Coefficient for Communication	-	0.82
Slope Coefficient for Process	-	1.87
R-squared	0.43	0.62
Significance F	0.18	0.01

The multiple linear regression-based analysis results show that the R-squared value was improved for both COSMIC and MicroM. The models constructed with the MicroM method represent adequate relationships considering the R-squared values.[99] On the other hand, although some improvements occurred, the models with the COSMIC method could not present a reliable relationship with the effort.

The evaluation results of the regression-based models are presented in Table 6.5. In this case, all models provided acceptable effort predictions, considering MMRE metric.[101] The MicroM models performed better accuracy than the COSMIC models, with a predictive accuracy (MMRE=0.18 for multiple linear regression). The results also indicate that multiple linear regression improved effort estimation accuracy for both methods. Furthermore, we also obtained the best PRED(30) value with a multiple linear regression-based MicroM model.

Table 6.5. Evaluation case study 2 results.

Method	Regression	MMRE	MdMRE	PRED(30)
COSMIC	Linear Regression	0.31	0.27	0.67
	Multiple Linear Regression	0.22	0.12	0.80
MicroM	Linear Regression	0.25	0.21	0.73
	Multiple Linear Regression	0.18	0.15	0.87

6.1.4. Evaluation Case Study 3

Organization 3 is an enterprise technology company that develops software solutions for the aviation domain. It has over 18 years of experience and more than 1500 employees. The projects included in the case study are in the aviation sector and are being developed using a traditional monolith architecture. The organization adopts Waterfall methodology and develops full-stack projects. The selected projects were completed previously, and the measurements were conducted based on requirement documents. The actual efforts are recorded on a project-by-project basis as person-day.

6.1.4.1. Execution of the Case Study

The case study began with an analysis of the requirement documents provided by the organization to evaluate whether they are measurable or not. We selected 14 project requirement documents, all of which have actual effort data, are completed, involve new development tasks, and have enough descriptions for measurement. In the first cycle, we measured the projects and constructed effort estimation models for both MicroM and COSMIC. In a later meeting with the organization, the assumptions made during the measurement process were clarified, and possible outlier projects were discussed. As a consensus of this meeting, we eliminated three outlier projects that could potentially affect the model's accuracy. Consequently, new effort estimation models were developed using the remaining 11 projects. The following section summarizes the result from the updated model constructed with 11 projects.

6.1.4.2. Results

We first performed linear regression analysis based on size measurement results. We used "total MicroM/COSMIC size" as the independent variable and "effort" as the dependent variable. Figure 6.3 shows the linear regression-based effort estimation models, including assessed residual plots, regression equations, and R-squared values.

The analysis results show that both COSMIC and MicroM methods demonstrate predictive relationships with actual effort.[99]

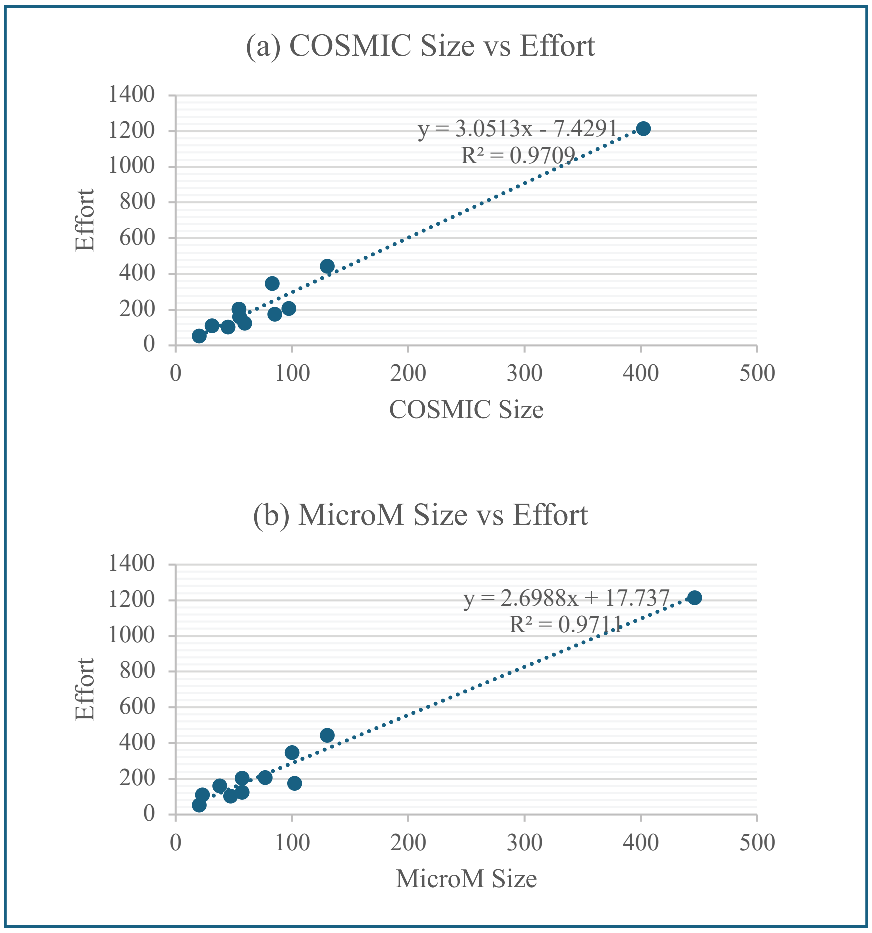

Figure 6.3. Linear regression-based effort estimation models for case study 3.

We performed multiple linear regression using three "MicroM event levels (Interaction, Communication, and Process) and four "COSMIC data movement types (Entry, Exit, Read, and Write)" as the independent variables and "actual effort" as the dependent variable. The multiple regression analysis results are presented in Table 6.6. The analysis results show that the R-squared values are still predictive for both COSMIC and MicroM.

The evaluation results of the regression-based models are presented in Table 6.7. The results show that all models provided acceptable effort predictions.[99] The COSMIC models performed better than the MicroM models, approaching the level of predictive accuracy with linear regression (MMRE=0.21 for linear regression). In contrast to the other case studies, we could not see any improvement with the MicroM method. Instead, COSMIC provided better effort estimation results. Here, we should note that Case Study 3 is designed as a counter case that involves projects developed in traditional architecture. Additionally, multiple linear regression-based models provided lower accuracy for both methods.

Table 6.6. Multiple linear regression analysis for evaluation case study 3.

	COSMIC	**MicroM**
Number of Observations	11	11
Intercept	6.38	20.39
Slope Coefficient for Entry	12.05	-
Slope Coefficient for Read	5.01	-
Slope Coefficient for Write	0.61	-
Slope Coefficient for Exit	-3.03	-
Slope Coefficient for Interaction	-	3.17
Slope Coefficient for Communication	-	5.22
Slope Coefficient for Process	-	-0.81
R-squared	0.98	0.97
Significance F	0.00	0.00

Table 6.7. Evaluation case study 3 results.

Method	**Regression**	**MMRE**	**MdMRE**	**PRED(30)**
COSMIC	Linear Regression	0.21	0.23	0.73
	Multiple Linear Regression	0.27	0.26	0.64
MicroM	Linear Regression	0.26	0.26	0.64
	Multiple Linear Regression	0.28	0.26	0.55

6.2. Discussion

This section begins by addressing the research questions posed for the evaluation case studies, followed by a discussion of additional findings.

6.2.1. Answering Evaluation Research Questions

The evaluation case studies aimed to assess whether the MicroM method developed for MSSA is a relevant size measurement technique. To achieve this, we measured projects from various organizations using both the MicroM and COSMIC methods across three distinct case studies and constructed effort estimation models using actual effort data. Additionally, we evaluated the relevance of MicroM, which was developed explicitly for MSSA, in a project implemented using a traditional monolithic architecture as a counter case. We used the R-squared value to assess the correlation between size and effort in simple linear and multiple linear regression analyses. We employed the MMRE, MdMRE, and PRED(30) metrics to evaluate the accuracy of the constructed effort estimation models.

We conducted Case Studies 1 and 2 to evaluate the relevance of the MicroM method for MSSA. In both case studies, the MicroM size provided a better R-squared value than the COSMIC size when correlated with actual effort. This was consistent for models created based on both the total number of events and different event levels. Furthermore, the results of both case studies demonstrated that the relationship between COSMIC size and effort is not reliable for planning.

In evaluating the accuracy of the effort estimation models, similar to the R-squared results, the models constructed using MicroM showed improvement over those developed using COSMIC. In Case Study 1, a 32% improvement was achieved (MMRE=0.32 vs. MMRE=0.22), while in Case Study 2, a 22% improvement was observed (MMRE=0.22 vs. MMRE=0.18). Despite significant improvements in both case studies, the greater improvement observed in Case Study 1 may be related to the type of the MS-based project. The project in Case Study 1 was event-driven, whereas the project in Case Study 2 was a request-driven MSSA. Since the MicroM method is based on counting events, it is expected to yield better results in event-based MSSA projects.

The multiple linear regression models, utilizing the event-level categories proposed by the MicroM method as input, demonstrated improvements in both case studies. In the first case study, this improvement was particularly observed in the MdMRE metric, while in the second case study, enhancements were noted in the MMRE, MdMRE, and PRED(30) metrics. These results suggest that utilizing multiple regression models, in addition to simple regression, in effort estimation with the MicroM method may lead to the development of more predictive models. In conclusion, both the total number of events and the event categories exhibited a stronger correlation with effort compared to the COSMIC method, resulting in a more predictive effort estimation model. This demonstrates the applicability of the MicroM method in MSSAs.

Finally, in Case Study 3, which was planned as a counter case involving a monolithic architecture, the R-squared values for both methods were very close and quite strong. However, the effort estimation models developed using the COSMIC method outperformed the models created with the MicroM method in both simple and multiple linear regression analyses. This result indicates that the COSMIC method continues to demonstrate success in monolithic architectures and confirms the MicroM method's relevance to MSSA.

6.2.2. Additional Findings

The evaluation case studies yielded additional findings, which are discussed as follows. The first additional finding is related to the type of measured items in Agile projects. In the first and second case studies, we observed that many of the issues and tasks were change requests, which emerged from the initial sprints of the project. Given the nature of Agile software development, where change requests are naturally part of sprints, we excluded these from our measurements. However, during discussions with both organizations, they indicated that they faced significant challenges in effort estimation related to change requests. In interviews conducted for this thesis, we identified that as the coupling between microservices increases in MSSA, the difficulties associated with bug fixes, testing, and change requests also rise. MicroM showed another utilization of software size in terms of events. The ratio of communication events in the MicroM method to the total number of events can be used to calculate coupling complexity in MSSA. High coupling complexity leads to many connections between microservices, which can cause management problems. Therefore, a change request related to any requirement will be more challenging to implement in a MS-based project with high coupling complexity. This finding can help software project management by improving decision-making for change request approvals, particularly by looking at change requests in terms of effort and complexity.

The second additional finding is related to the varying effects of event levels on effort in MS-based projects. In Case Studies 1 and 2, we found that functional, architectural, and algorithmic level events affect effort differently. In particular, algorithmic level events significantly impacted effort more than the other levels. When we discussed this finding with the organizations, they argued that tasks related to front-end and database operations at the functional level, as well as communication between microservices at the architectural level, have become standard in their projects. However, algorithmic-level tasks vary based on projects. Algorithmic-level tasks involve understanding the project's business rules correctly. Organization representatives stated that they often discuss the business rules related to the project and that these tasks require more effort. This situation may vary in other organizations, where the effects of functional or architectural-level events on effort could be more significant. Therefore, the MSSA-based event level categorization of MicroM, combined with multiple regression, will be a useful practice for improving effort estimation accuracy in organizations.

In the MicroM method, the communication events at the architectural level can be argued to be similar to the COSMIC method's Entry and Exit data movements. An event consumed by a microservice from a queue can be seen as an Entry, while an event published by the microservice can be seen as an Exit data movement. However, it is essential to define these events at different abstraction levels. User interaction and communication between microservices are not at the same abstraction level. User interaction can be determined during the requirements phase, while communication between microservices is defined during the design phase. Additionally, we observed that the effects of the abstraction levels in the MicroM method on effort are different. In the MicroM method, the architectural level was explicitly defined for MSSA, but these events can adapt based on other architectures.

The final additional finding relates to the eEPC modeling notation proposed by the MicroM method for identifying events. It can be discussed that UML notations can also be used to identify events. For example, in a UML sequence diagram, communication occurs through objects. Therefore, this approach does not facilitate event identification. On the other hand, in an eEPC diagram, requirements are defined through events, making it better aligned with MSSA.

6.3. Threats to Validity

Yin[88] identifies four types of validity threats in case studies: construct validity, external validity, internal validity, and reliability. Since this thesis involves multiple case studies for evaluation, we applied Yin's framework for addressing these threats. In this section, we discuss the potential threats to the validity and how we mitigated them.

Construct validity deals with the question whether appropriate operational measures are established for the concepts under research. Yin recommends using multiple sources of evidence to ensure construct validity. In our evaluation case studies, we utilized projects from three distinct software development companies, encompassing various domains and developed using different methodologies, each with unique requirement formats and levels of detail.

External validity addresses the generalizability of findings. According to Yin, single-case studies are used to develop theory, while multiple-case studies are employed to test its replication across different contexts. In this thesis, we performed single-case studies in real-world software organizations during the problem identification and solution design phases, followed by multiple-case studies during the evaluation phase. For the evaluation, we performed multiple case studies on projects from three different organizations, distinct from those involved in the problem identification and solution design phases, to increase the generalizability of findings. For the evaluation, we conducted multiple case studies on projects from three different organizations, distinct from those involved in the problem identification and solution design phases, to enhance the generalizability of the findings. The evaluation included measurements performed on projects covering three different domains -finance, MIS, and aviation- using three different architectures: event-driven MSSA, request-driven MSSA, and monolith. The projects also featured two types of software development -backend and full stack- along with two levels of documentation at the requirement and design stages. Additionally, the measured artifacts included three categories: issues, tasks, and requirement documents.

Internal validity is concerned with the challenge of establishing a causal relationship whereby certain conditions are shown to lead other conditions as distinguished from spurious relationships. In this thesis, we developed the MicroM method for measuring size specifically tailored for microservices. To validate this objective, we evaluated the success of the MicroM method in two case studies involving microservice-based architecture, and as a counter case, we conducted a third case study on a traditional monolithic project. MicroM demonstrated improvements in Case Studies 1 and 2, while it did not yield any improvements in Case Study 3. Another threat to internal validity may arise from the limited sample size from a statistical analysis perspective. However, in the field of software engineering, such conditions are observed due to the inherent challenges of obtaining practical and large samples.[32,52,97,112] Abran[113] argues that high correlation values in small samples do not necessarily represent a valid statistical reality and that these results must be interpreted with caution. Furthermore, he emphasizes that models developed using small sample sizes can still be valuable and useful for the organization where the data was collected, serving as indicators for projects conducted under similar conditions. In this thesis, we evaluated the validity of the method we developed by examining its relationship with effort. However, other factors may also influence effort in software projects. To mitigate this, we selected issues and tasks completed within the same team for Case 1 and Case 2 during the measurement process, ensuring that the developers implemented these requirements were at the same skill level. For Case Study 3, we measured projects belonging to teams with similar levels.

Reliability refers to the extent to which a study can be repeated with consistent data collection methods, yielding similar results. Yin suggests ensuring reliability in case studies by adhering to a defined protocol and creating a case study database. In this thesis, we first developed a case study plan and established case selection criteria for all case studies. We conducted the studies according to the defined plan and selected the cases based on the predetermined criteria. In the conducted case studies, the reliability of the measurements could pose a threat to validity. Although the MicroM and COSMIC size measurement methods are based on objective rules, when applying these rules to real-life projects. Although the measurement rules are well-defined, measurers may make assumptions when the requirements are not adequately specified. To mitigate potential threats related to this human judgment, we established criteria in the case selection process to ensure that organizations were accessible when needed. We completed our measurements in two phases. After the initial phase, the assumptions made by the measurer were discussed in a meeting with the organization, allowing for necessary adjustments. Any unclear measurements were revisited in collaboration with representatives from the organization. As a result, precautions were taken to ensure accurate measurements. For future MicroM measurements to be conducted by various researchers, this thesis provides a detailed, step-by-step explanation of the method, supported by examples. In this way, other researchers who want to repeat this research can reach similar results using these procedures.

6.4. Summary of the Chapter

This chapter presented the three case studies conducted to evaluate the MicroM method and discussed the findings. The evaluation results showed that the MicroM method correlates more strongly with effort than the COSMIC method in MS-based projects. MicroM achieved an improvement of up to 32% in Mean Magnitude of Relative Error (MMRE) in the effort estimation models compared to the COSMIC FSM method. The multiple linear regression models, utilizing the event-level categories proposed by the MicroM method as input, also demonstrated improvements in effort estimation accuracy. In addition, it was observed that the impact of algorithmic-level events on effort is more significant than that of other levels. Finally, it was found that the COSMIC method provided a more accurate estimation of effort for projects developed in traditional monolithic architecture. This result indicated that the COSMIC method continues to demonstrate success in monolithic architectures and confirmed the MicroM method's relevance to MSSA.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

With the paradigm shift in today's software industry, MSSA has emerged as a preferred design approach for a growing number of organizations. MSSA has shifted from a data-driven to a behavior-oriented structure, where the traditional single database is replaced by independent microservices, each with its own database. Every service manages its own data, designed to be isolated, scalable, and fault tolerant. These services work together to form a unified system, offering much greater flexibility than traditional monolithic systems.

In software projects, meeting customer expectations on time and within budget is achieved through effective project management, particularly with accurate effort estimation. FSM methods have long been used for objective size measurement in monolithic architectures. However, these methods are often inadequate for measuring the size of modern MS-based applications. Unlike monolithic architectures that rely on data flow through functions (such as method parameters), MSSA employs structures like event queues for communication. MSSA requires a behavior-focused approach using service calls (e.g., REST API), making the relational data-based FSM methods less effective for accurate effort or cost estimation in MSSA projects.

The literature review revealed a lack of research on size measurement methods tailored to MSSA. Surveys and interviews we conducted confirmed that there is no de facto method in the industry for sizing MS-based projects. Additionally, organizations face distinct challenges in size measurement and effort estimation for such projects. These findings highlighted the need to develop a new size measurement method specifically designed for MSSA: MicroM.

This thesis aimed to create an objective size measurement method specifically for MSSA, using the DSR methodology to develop MicroM. An iterative and incremental approach was applied within the DSR process, drawing inspiration from events, the base components of microservices. We first conducted an exploratory case study to assess the effectiveness of event-based size measurement and found that it performed better than the data movement-counting FSM methods in MSSA. As a result, we developed MicroM method, considering the characteristics of MSSA and the limitations of existing FSM methods.

MicroM focuses on categorizing events at three abstraction levels of the software representation based on architectural characteristics: (1) Functional Level, (2) Architectural Level, and (3) Algorithmic Level. These levels allow MicroM to be used throughout the SDLC, from the requirements phase to post-project stages. The Functional Level covers user and database interaction events, the Architectural Level focuses on internal system communication events, and the Algorithmic Level addresses processing events such as decision-making and calculations. This detailed approach complements existing FSM methods and provides a more precise size measurement for MSSA.

Three case studies were conducted to evaluate MicroM's success and reliability. The results showed that MicroM had a stronger correlation with effort compared to the COSMIC FSM method in MS-based projects, improving effort estimation accuracy by up to 32% in Mean Magnitude of Relative Error (MMRE). Multiple linear regression models using MicroM's event-level categories further enhanced effort estimation accuracy. However, COSMIC performed better in monolithic architectures, confirming that while COSMIC remains effective for traditional systems, MicroM is better suited for MSSA. Consequently, this thesis has led to the identification of the following research prospects:

- In the evaluation case studies, we measured size in MS-based projects using small items such as tasks and issues and developed effort estimation models. The variance between the measured items was minimal. In other words, the size and effort of the measured requirements were very close to each other. Future studies could focus on effort estimation using larger items, such as sprints or other grouped elements. Additionally, in these case studies, we used tasks and issues developed by multiple developers as input for the same effort estimation model. Future studies could focus on developer-specific effort estimation models by constructing distinct effort estimation models for each developer. Both approaches may improve the accuracy of effort estimations.
- In this thesis, we evaluated the method in three case studies from three different domains to ensure the generalizability of the results. These case studies represented three different domains, two software architectures, two software development types, two levels of documentation, and three types of measured

artifacts. To further improve the generalizability, additional case studies could be conducted in future research, such as organizations in different domains.

- This thesis emphasized the critical role of change requests in the software development lifecycle, particularly within Agile methodologies. Future research could focus on developing a method designed to measure the size of change requests. In such a method, the coupling complexity framework established by the MicroM method may serve as a valuable foundation.
- The interviews conducted in this thesis revealed that organizations often struggle to allocate resources and expert workforce for objective size measurement. Therefore, developing automated tools that can measure size using both the MicroM method and other objective size measurement approaches would represent a valuable avenue for future research. In today's context, advanced artificial intelligence models could be leveraged for this purpose.

REFERENCES

1. Di Francesco, P.; Lago, P.; Malavolta, I. Architecting with Microservices: A Systematic Mapping Study. Journal of Systems and Software 2019, 150, 77–97. https://doi.org/10.1016/j.jss.2019.01.001.
2. Larrucea, X.; Santamaria, I.; Colomo-Palacios, R.; Ebert, C. Microservices. IEEE Software 2018, 35 (3), 96–100. https://doi.org/10.1109/MS.2018.2141030.
3. Bonér, J. Reactive Microservices Architecture; O'Reilly Media, Inc., 2016.
4. Thönes, J. Microservices. IEEE Software 2015, 32 (1), 116–116. https://doi.org/10.1109/MS.2015.11.
5. Sampaio, A. R.; Kadiyala, H.; Hu, B.; Steinbacher, J.; Erwin, T.; Rosa, N.; Beschastnikh, I.; Rubin, J. Supporting Microservice Evolution. In 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME); 2017; pp 539–543. https://doi.org/10.1109/ICSME.2017.63.
6. Dragoni, N.; Giallorenzo, S.; Lafuente, A. L.; Mazzara, M.; Montesi, F.; Mustafin, R.; Safina, L. Microservices: Yesterday, Today, and Tomorrow. In Present and Ulterior Software Engineering; Mazzara, M., Meyer, B., Eds.; Springer International Publishing: Cham, 2017; pp 195–216. https://doi.org/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 2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA); 2021; pp 244–251. https://doi.org/10.1109/SEAA53835.2021.00038.
8. Bonér, J. *Reactive Microsystems*; O'Reilly Media, Inc., 2017.
9. Rolfe, D. *What is an Event-Driven Microservices Architecture?*. Volt Active Data. https://www.voltactivedata.com/blog/2022/10/what-is-event-driven-microservices-architecture/ (accessed 2023-05-13).
10. Newman, S. *Building Microservices*; O'Reilly Media, Inc., 2021.
11. Fowler, M.; Highsmith, J. The Agile Manifesto. *Software development* 2001, *9* (8), 28–35.
12. Garousi, V.; Coşkunçay, A.; Betin-Can, A.; Demirörs, O. A Survey of Software Engineering Practices in Turkey. Journal of Systems and Software 2015, 108, 148–177. https://doi.org/10.1016/j.jss.2015.06.036.

13. *Standish Group Chaos Report*; T.S.G. International, 2020.
14. Jørgensen, M.; Boehm, B.; Rifkin, S. Software Development Effort Estimation: Formal Models or Expert Judgment? *IEEE software* 2009, *26* (2), 14–19.
15. Usman, M.; Mendes, E.; Weidt, F.; Britto, R. Effort Estimation in Agile Software Development: A Systematic Literature Review. In *Proceedings of the 10th International Conference on Predictive Models in Software Engineering*; PROMISE '14; Association for Computing Machinery: New York, NY, USA, 2014; pp 82–91. https://doi.org/10.1145/2639490.2639503.
16. Hacaloğlu, T.; Demirörs, O. Challenges of Using Software Size in Agile Software Development: A Systematic Literature Review. *Academic Papers at IWSM Mensura 2018* 2018.
17. Hacaloglu, T.; Demirors, O. Measureability of Functional Size in Agile Software Projects: Multiple Case Studies with COSMIC FSM. In *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*; 2019; pp 204–211. https://doi.org/10.1109/SEAA.2019.00041.
18. Commeyne, C.; Abran, A.; Djouab, R. Effort Estimation with Story Points and Cosmic Function Points-an Industry Case Study. *Software Measurement News* 2016, *21* (1), 25–36.
19. Kang, S.; Choi, O.; Baik, J. Model-Based Dynamic Cost Estimation and Tracking Method for Agile Software Development. In *2010 IEEE/ACIS 9th International Conference on Computer and Information Science*; 2010; pp 743–748. https://doi.org/10.1109/ICIS.2010.126.
20. Hohman, M. M. Estimating in Actual Time [Extreme Programming]. In *Agile Development Conference (ADC'05)*; 2005; pp 132–138. https://doi.org/10.1109/ADC.2005.22.
21. Huijgens, H.; Solingen, R. van. A Replicated Study on Correlating Agile Team Velocity Measured in Function and Story Points. In *Proceedings of the 5th International Workshop on Emerging Trends in Software Metrics*; WETSoM 2014; Association for Computing Machinery: New York, NY, USA, 2014; pp 30–36. https://doi.org/10.1145/2593868.2593874.
22. Buglione, L.; Trudel, S. Guideline for Sizing Agile Projects with COSMIC. *Proceedings of the IWSM/MetriKon/Mensura, Stuttgart, Germany* 2010.

23. Ozkan, B.; Turetken, O.; Demirors, O. Software Functional Size: For Cost Estimation and More. In *European Conference on Software Process Improvement*; Springer, 2008; pp 59–69.
24. Czarnacka-Chrobot, B. Standardization of Software Size Measurement. In *Internet – Technical Development and Applications*; Tkacz, E., Kapczynski, A., Eds.; Advances in Intelligent and Soft Computing; Springer: Berlin, Heidelberg, 2009; pp 149–156. https://doi.org/10.1007/978-3-642-05019-0_17.
25. Gencel, C.; Demirors, O. Functional Size Measurement Revisited. *ACM Trans. Softw. Eng. Methodol.* 2008, *17* (3), 15:1-15:36. https://doi.org/10.1145/1363102.1363106.
26. *ISO/IEC 20926:2009 - IFPUG functional size measurement method 2009*. ISO. https://www.iso.org/standard/51717.html (accessed 2023-06-06).
27. *ISO/IEC 29881:2010 - FiSMA 1.1 functional size measurement method*. ISO. https://www.iso.org/standard/56418.html (accessed 2023-06-06).
28. *ISO/IEC 24570:2018 -NESMA functional size measurement method*. ISO. https://www.iso.org/standard/72505.html (accessed 2023-06-06).
29. ISO/IEC 20968:2002 - Software Engineering — Mk II Function Point Analysis. https://www.iso.org/standard/35603.html (accessed 2024-10-02).
30. *ISO/IEC 19761:2017 - Software Engineering – COSMIC: A Functional Size Measurement Method*; International Organization for Standardization, 2017.
31. Ungan, E.; Çizmeli, N.; Demirörs, O. Comparison of Functional Size Based Estimation and Story Points, Based on Effort Estimation Effectiveness in SCRUM Projects. In *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*; 2014; pp 77–80. https://doi.org/10.1109/SEAA.2014.83.
32. Salmanoglu, M.; Hacaloglu, T.; Demirors, O. Effort Estimation for Agile Software Development: Comparative Case Studies Using COSMIC Functional Size Measurement and Story Points. In *Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement*; IWSM Mensura '17; Association for Computing Machinery: Gothenburg, Sweden, 2017; pp 41–49. https://doi.org/10.1145/3143434.3143450.
33. Ünlü, H.; Hacaloglu, T.; Büber, F.; Berrak, K.; Leblebici, O.; Demirörs, O. Utilization of Three Software Size Measures for Effort Estimation in Agile World: A Case Study. In *2022 48th Euromicro Conference on Software Engineering and*

Advanced Applications (SEAA); 2022; pp 239–246. https://doi.org/10.1109/SEAA56994.2022.00045.

34. *COSMIC Measurement Manual Version 5.0*; The Common Software Measurement International Consortium, 2021. https://cosmic-sizing.org/publications/measurement-manual-v4-0-2/.
35. Banerjee, S.; Sarkar, A. Modeling NoSQL Databases: From Conceptual to Logical Level Design. In *3rd International Conference Applications and Innovations in Mobile Computing (AIMoC 2016), Kolkata, India, February*; 2016; pp 10–12.
36. Sucaciu, B. *How event-driven architecture solves modern web app problems*. Stack Overflow Blog. https://stackoverflow.blog/2020/03/16/how-event-driven-architecture-solves-modern-web-app-problems/ (accessed 2021-09-22).
37. Ünlü, H.; Kennouche, D. E.; Soylu, G. K.; Demirörs, O. Microservice-Based Projects in Agile World: A Structured Interview. *Information and Software Technology* 2024, *165*, 107334. https://doi.org/10.1016/j.infsof.2023.107334.
38. Ünlü, H.; Bilgin, B.; Demirors, O. A Survey on Organizational Choices for Microservice-Based Software Architectures. *Turkish Journal of Electrical Engineering and Computer Sciences* 2022, *30* (4), 1187–1203. https://doi.org/10.55730/1300-0632.3843.
39. Hevner, A. R.; March, S. T.; Park, J.; Ram, S. Design Science in Information Systems Research. *MIS quarterly* 2004, 75–105.
40. Offermann, P.; Levina, O.; Schönherr, M.; Bub, U. Outline of a Design Science Research Process. In *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*; DESRIST ’09; Association for Computing Machinery: New York, NY, USA, 2009; pp 1–11. https://doi.org/10.1145/1555619.1555629.
41. Rodgers, P. Service-Oriented Development on Netkernel-Patterns, Processes & Products to Reduce System Complexity. In *CloudComputingExpo*; 2005.
42. Fowler, M.; Lewis. *Microservices*. martinfowler.com. https://martinfowler.com/articles/microservices.html (accessed 2021-03-11).
43. Merkel, D. Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux j* 2014, *239* (2).
44. Ihuman. *RESTful vs Event-Driven in Microservices: Key Differences Explained*. Ambassador. https://www.getambassador.io/blog/request-driven-restful-vs-event-driven-in-microservices (accessed 2024-10-07).

45. Fögen, K. *Event-driven Microservices with Request/Response APIs*. Thoughtworks. https://www.thoughtworks.com/insights/blog/apis/event-driven-microservices-with-request-part-one (accessed 2024-10-07).
46. Demirors, O. *CENG 555 - Analysis and Design of Microservice Based Systems*. https://ceng.iyte.edu.tr/courses/ceng-555/.
47. Albrecht, A. J. Measuring Application Development Productivity. In *Proceedings of IBM Applications Development Symposium*; Monterey, 1979; Vol. 83, pp 14–17.
48. *Guideline for Sizing Business Application Software*; The Common Software Measurement International Consortium, 2017.
49. *Guideline for Sizing Real-Time Software*; The Common Software Measurement International Consortium, 2015.
50. *Guideline for Sizing Service-Oriented Architecture Software*; The Common Software Measurement International Consortium, 2015.
51. Hacaloglu, T. Event Points: A Software Size Measurement Model, Middle East Technical University, Ankara, 2021.
52. Hacaloglu, T.; Demirors, O. An Exploratory Case Study Using Events as a Software Size Measure. *Inf Technol Manag* 2023. https://doi.org/10.1007/s10799-023-00394-y.
53. Scheer, A.-W.; Thomas, O.; Adam, O. Process Modeling Using Event-Driven Process Chains. In *Process-Aware Information Systems*; John Wiley & Sons, Ltd, 2005; pp 119–145. https://doi.org/10.1002/0471741442.ch6.
54. van der Aalst, W. M. P. Formalization and Verification of Event-Driven Process Chains. *Information and Software Technology* 1999, *41* (10), 639–650. https://doi.org/10.1016/S0950-5849(99)00016-6.
55. Dragicevic, S.; Celar, S.; Novak, L. Use of Method for Elicitation, Documentation, and Validation of Software User Requirements (MEDoV) in Agile Software Development Projects. In *2014 Sixth International Conference on Computational Intelligence, Communication Systems and Networks*; 2014; pp 65–70. https://doi.org/10.1109/CICSyN.2014.27.
56. Riehle, D. M.; Jannaber, S.; Karhof, A.; Thomas, O.; Delfmann, P.; Becker, J. On the De-Facto Standard of Event-Driven Process Chains: How EPC Is Defined in Literature; Gesellschaft für Informatik e.V., 2016; pp 61–76.
57. (Demirors, O. CENG 323 - Project Management - Lecture 2 - Problem Analysis, 2024. https://ceng.iyte.edu.tr/courses/ceng-323/.

58. Lübke, D. Transformation of Use Cases to EPC Models. In *5. Workshop der Gesellschaft für Informatik e.V. (GI) und Treffen ihres Arbeitskreises "Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (WI-EPK)*; CEUR Workshop Proceedings: Vienna, austria, 2006; pp 137–156.
59. Gross, A.; Doerr, J. EPC vs. UML Activity Diagram - Two Experiments Examining Their Usefulness for Requirements Engineering. In *2009 17th IEEE International Requirements Engineering Conference*; 2009; pp 47–56. https://doi.org/10.1109/RE.2009.30.
60. Amjad, A.; Azam, F.; Anwar, M. W.; Butt, W. H.; Rashid, M. Event-Driven Process Chain for Modeling and Verification of Business Requirements–A Systematic Literature Review. *IEEE Access* 2018, *6*, 9027–9048. https://doi.org/10.1109/ACCESS.2018.2791666.
61. Asik, T.; Selcuk, Y. E. Policy Enforcement upon Software Based on Microservice Architecture. In *2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA)*; 2017; pp 283–287. https://doi.org/10.1109/SERA.2017.7965739.
62. Vural, H.; Koyuncu, M.; Misra, S. A Case Study on Measuring the Size of Microservices. In *Computational Science and Its Applications – ICCSA 2018*; Gervasi, O., Murgante, B., Misra, S., Stankova, E., Torre, C. M., Rocha, A. M. A. C., Taniar, D., Apduhan, B. O., Tarantino, E., Ryu, Y., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, 2018; pp 454–463. https://doi.org/10.1007/978-3-319-95174-4_36.
63. Taibi, D.; Systä, K. A Decomposition and Metric-Based Evaluation Framework for Microservices. In *Cloud Computing and Services Science*; Ferguson, D., Méndez Muñoz, V., Pahl, C., Helfert, M., Eds.; Communications in Computer and Information Science; Springer International Publishing: Cham, 2020; pp 133–149. https://doi.org/10.1007/978-3-030-49432-2_7.
64. IFPUG. *White Paper: Applying FPA to Microservices*; 2022. https://ifpug.mclms.net/en/package/10193/course/19545/view (accessed 2024-10-07).
65. Schneider, S. *Events*. Internet Encyclopedia of Philosophy. https://iep.utm.edu/events/ (accessed 2024-10-05).
66. Kim, J. *Supervenience and Mind: Selected Philosophical Essays*; Cambridge University Press, 1993.

67. Davidson, D.; Pollock, F. *Essays on Actions and Events*; 2001.
68. Lewis, D. *Counterfactuals*; John Wiley & Sons, 2013.
69. Lewis, D. New Work for a Theory of Universals. *Australasian Journal of Philosophy* 1983. https://doi.org/10.1080/00048408312341131.
70. Lewis, D. *Philosophical Papers*; Oxford University Press, 1983.
71. Lewis, D. *On the Plurality of Worlds*; 1986.
72. Aarab, Z.; Saidi, R.; Rahmani, M. D. Event-Driven Modeling for Context-Aware Information Systems. In *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*; 2016; pp 1–8. https://doi.org/10.1109/AICCSA.2016.7945785.
73. Kappel, G.; Pröll, B.; Retschitzegger, W.; Schwinger, W. Modelling Ubiquitous Web Applications - The WUML Approach. In *Conceptual Modeling for New Information Systems Technologies*; Arisawa, H., Kambayashi, Y., Kumar, V., Mayr, H. C., Hunt, I., Eds.; Springer: Berlin, Heidelberg, 2002; pp 183–197. https://doi.org/10.1007/3-540-46140-X_15.
74. Kong, J.; Jung, J.-Y.; Park, J. Event-Driven Service Coordination for Business Process Integration in Ubiquitous Enterprises. *Computers & Industrial Engineering* 2009, *57* (1), 14–26. https://doi.org/10.1016/j.cie.2008.08.019.
75. Fowler, M. *Domain Event*. martinfowler.com. https://martinfowler.com/eaaDev/DomainEvent.html (accessed 2024-10-05).
76. Vernon, V. *Implementing Domain-Driven Design*; Addison-Wesley, 2013.
77. Davis, R. *Business Process Modelling with ARIS: A Practical Guide*; Springer Science & Business Media, 2001.
78. Afshar, S.; Ralph, N.; Xu, Y.; Tapson, J.; Schaik, A. van; Cohen, G. Event-Based Feature Extraction Using Adaptive Selection Thresholds. *Sensors* 2020, *20* (6), 1600. https://doi.org/10.3390/s20061600.
79. Chima, R. *Blog: Event-Driven Applications In Software Development*. Blueberry Custom Software. https://www.bbconsult.co.uk/blog/event-driven-applications/ (accessed 2024-10-05).
80. Bögl, A.; Kobler, M.; Schrefl, M. Knowledge Acquisition from EPC Models for Extraction of Process Patterns in Engineering Domains. In *Multikonferenz Wirtschaftsinformatik*; 2008.

81. Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M. C.; Regnell, B.; Wesslén, A. *Experimentation in Software Engineering*; Springer Science & Business Media, 2012.
82. Rabhi, F.; Demirörs, O. *Software Engineering Research - White Paper*; 2018.
83. Runeson, P.; Engström, E.; Storey, M.-A. The Design Science Paradigm as a Frame for Empirical Software Engineering. In *Contemporary Empirical Methods in Software Engineering*; Felderer, M., Travassos, G. H., Eds.; Springer International Publishing: Cham, 2020; pp 127–147. https://doi.org/10.1007/978-3-030-32489-6_5.
84. Wieringa, R. J. *Design Science Methodology for Information Systems and Software Engineering*; Springer, 2014.
85. Wohlin, C.; Höst, M.; Henningsson, K. Empirical Research Methods in Software Engineering. In *Empirical Methods and Studies in Software Engineering: Experiences from ESERNET*; Conradi, R., Wang, A. I., Eds.; Lecture Notes in Computer Science; Springer: Berlin, Heidelberg, 2003; pp 7–23. https://doi.org/10.1007/978-3-540-45143-3_2.
86. Shull, F.; Singer, J.; Sjøberg, D. I. *Guide to Advanced Empirical Software Engineering*; Springer, 2007.
87. Linaker, J.; Sulaman, S. M.; Höst, M.; de Mello, R. M. Guidelines for Conducting Surveys in Software Engineering v. 1.1. *Lund University* 2015.
88. Yin Robert, K. *Case Study Research and Applications: Design and Methods*; Los Angeles, CA: Sage Publications, 2017.
89. Zainal, Z. Case Study as a Research Method. *Jurnal kemanusiaan* 2007, *5* (1).
90. Runeson, P.; Höst, M. Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *Empir Software Eng* 2008, *14* (2), 131. https://doi.org/10.1007/s10664-008-9102-8.
91. Malhotra, R. *Empirical Research in Software Engineering: Concepts, Analysis, and Applications*; CRC Press, 2016.
92. Perry, D. E.; Sim, S. E.; Easterbrook, S. M. Case Studies for Software Engineers. In *Proceedings. 26th International Conference on Software Engineering*; 2004; pp 736–738. https://doi.org/10.1109/ICSE.2004.1317512.
93. Easterbrook, S.; Singer, J.; Storey, M.-A.; Damian, D. Selecting Empirical Methods for Software Engineering Research. In *Guide to Advanced Empirical Software*

Engineering; Shull, F., Singer, J., Sjøberg, D. I. K., Eds.; Springer: London, 2008; pp 285–311. https://doi.org/10.1007/978-1-84800-044-5_11.

94. Runeson, P.; Host, M.; Rainer, A.; Regnell, B. *Case Study Research in Software Engineering: Guidelines and Examples*; John Wiley & Sons, 2012.
95. Sykes, A. o. *An Introduction to Regression Analysis*; 1993.
96. Montgomery, D. C.; Peck, E. A.; Vining, G. G. *Introduction to Linear Regression Analysis*; John Wiley & Sons, 2021.
97. Abran, A.; Desharnais, J.-M.; Zarour, M.; Demirörs, O. Productivity-Based Software Estimation Models and Process Improvement: An Empirical Study. *Int. J. Adv. Softw* 2015, *8* (1 & 2), 103–114.
98. Ünlü, H.; Yalçın, A. G.; Öztürk, D.; Akkaya, G.; Kalecik, M.; Ekici, N. U.; Orhan, O.; Çiftçi, O.; Yumlu, S.; Demirörs, O. Software Effort Estimation Using ISBSG Dataset: Multiple Case Studies. In *2021 15th Turkish National Software Engineering Symposium (UYMS)*; 2021; pp 1–6. https://doi.org/10.1109/UYMS54260.2021.9659655.
99. Humphrey, W. S. *A Discipline for Software Engineering*; Pearson Education India, 1995.
100. Conte, S. D.; Dunsmore, H. E.; Shen, Y. E. Software Engineering Metrics and Models; Benjamin-Cummings Publishing Co., Inc.: USA, 1986.
101. Hastings, T. E.; Sajeev, A. S. M. A Vector-Based Approach to Software Size Measurement and Effort Estimation. IEEE Transactions on Software Engineering 2001, 27 (4), 337–350. https://doi.org/10.1109/32.917523.
102. Foss, T.; Stensrud, E.; Kitchenham, B.; Myrtveit, I. A Simulation Study of the Model Evaluation Criterion MMRE. *IEEE Transactions on Software Engineering* 2003, *29* (11), 985–995. https://doi.org/10.1109/TSE.2003.1245300.
103. Kitchenham, B. A.; Pickard, L. M.; MacDonell, S. G.; Shepperd, M. J. What Accuracy Statistics Really Measure. *IEE Proceedings - Software* 2001, *148* (3), 81–85. https://doi.org/10.1049/ip-sen:20010506.
104. Myrtveit, I.; Stensrud, E.; Shepperd, M. Reliability and Validity in Comparative Studies of Software Prediction Models. *IEEE Transactions on Software Engineering* 2005, *31* (5), 380–391. https://doi.org/10.1109/TSE.2005.58.
105. Jorgensen, M. Experience with the Accuracy of Software Maintenance Task Effort Prediction Models. *IEEE Transactions on Software Engineering* 1995, *21* (8), 674–681. https://doi.org/10.1109/32.403791.

106. Jørgensen, M.; Halkjelsvik, T.; Liestøl, K. When Should We (Not) Use the Mean Magnitude of Relative Error (MMRE) as an Error Measure in Software Development Effort Estimation? *Information and Software Technology* 2022, *143*, 106784. https://doi.org/10.1016/j.infsof.2021.106784.
107. Jeffery, R.; Ruhe, M.; Wieczorek, I. A Comparative Study of Two Software Development Cost Modeling Techniques Using Multi-Organizational and Company-Specific Data. *Information and Software Technology* 2000, *42* (14), 1009–1016. https://doi.org/10.1016/S0950-5849(00)00153-1.
108. van Koten, C.; Gray, A. R. An Application of Bayesian Network for Predicting Object-Oriented Software Maintainability. *Information and Software Technology* 2006, *48* (1), 59–67. https://doi.org/10.1016/j.infsof.2005.03.002.
109. MacDonell, S. G. Establishing Relationships between Specification Size and Software Process Effort in CASE Environments. Information and Software Technology 1997, 39 (1), 35–45. https://doi.org/10.1016/0950-5849(96)01125-1.
110. MacDonell, S. G.; Gray, A. R. A comparison of modeling techniques for software development effort prediction. 1998.
111. Ünlü, H.; Hacaloğlu, T.; Ömüral, N. K.; Çalişkanel, N.; Leblebici, O.; Demirörs, O. An Exploratory Case Study on Effort Estimation in Microservices. In *2023 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*; 2023; pp 215–218. https://doi.org/10.1109/SEAA60479.2023.00040.
112. Yürüm, O. R.; Ünlü, H.; Demirörs, O. Towards the Construction of a Software Benchmarking Dataset via Systematic Literature Review. In *The Joint Conference of the 32nd International Workshop on Software Measurement (IWSM) and the 17th International Conference on Software Process and Product Measurement (MENSURA)*; 2023.
113. Abran, A. *Software Project Estimation: The Fundamentals for Providing High Quality Information to Decision Makers*; John Wiley & Sons, 2015.

APPENDIX

DETAILED RESULTS OF THE CASE STUDIES

Table A.1. Exploratory case study 1 detailed results - COSMIC and Event Point (Eff-Ac: Actual effort in person-hour, Eff-Pr: Predicted Effort in person-hour).

Item	COSMIC	Event Point	Eff-Ac	COSMIC		Event Point	
No	Size	Size		Eff-Pr	MRE	Eff-Pr	MRE
1	53	64	8.84	7.60	0.36	10.47	0.12
2	6	6	3.00	1.50	0.49	1.25	0.58
3	49	20	3.01	7.08	1.76	3.48	0.36
4	37	30	7.21	5.52	0.26	5.07	0.32
5	6	6	0.67	1.50	0.42	1.25	0.19
6	3	3	2.10	1.11	0.49	0.77	0.64
7	21	35	3.17	3.44	0.33	5.86	1.26
8	6	7	2.00	1.50	0.00	1.41	0.05
9	23	21	2.40	3.70	0.98	3.64	0.94
10	6	11	1.80	1.50	0.20	2.05	0.10
11	27	17	4.44	4.22	0.16	3.00	0.18
12	12	10	2.14	2.27	0.20	1.89	0.00
13	6	16	2.41	1.50	0.31	2.84	0.31
14	12	9	1.30	2.27	1.13	1.73	0.61
				MMRE	0.51	**MMRE**	0.41
				MdMRE	0.35	**MdMRE**	0.32
				PRED(30)	0.31	**PRED(30)**	0.43

Table A.2. Exploratory case study 2 detailed results - COSMIC (Eff-Ac: Actual effort in person-hour, Eff-Pr: Predicted Effort in person-hour, Lin. Reg: Linear Regression, Mult. Lin. Reg: Multiple Linear Regression).

Item	**COSMIC Size**					**Eff-Ac**	**Lin. Reg.**		**Mult. Lin. Reg.**	
No	**Entry**	**Read**	**Write**	**Exit**	**Total**		**Eff-Pr**	**MRE**	**Eff-Pr**	**MRE**
1	2	0	0	2	4	2.50	6.43	1.57	7.45	1.98
2	1	0	1	1	3	2.50	5.73	1.29	5.79	1.32
3	2	2	0	2	6	4.00	7.83	0.96	6.84	0.71
4	1	1	0	2	4	4.00	6.43	0.61	6.90	0.72
5	2	1	1	1	5	4.50	7.13	0.58	5.73	0.27
6	2	0	0	2	4	6.00	6.43	0.07	7.45	0.24
7	3	2	2	2	9	6.50	9.92	0.35	8.49	0.31
8	1	1	0	1	3	7.00	5.73	0.18	4.78	0.32
9	2	2	1	2	7	8.00	8.53	0.07	7.54	0.06
10	4	0	0	3	7	8.00	8.53	0.07	10.07	0.26
11	1	1	0	1	3	10.00	5.73	0.43	4.78	0.52
12	2	2	0	2	6	11.50	7.83	0.32	6.84	0.41
13	4	4	0	4	12	13.00	12.02	0.08	10.95	0.16
14	2	0	2	3	7	15.00	8.53	0.43	10.97	0.27
15	9	8	3	8	28	18.00	23.20	0.29	21.54	0.20
16	7	2	1	7	17	20.00	15.52	0.22	19.37	0.03
17	7	3	2	5	17	20.50	15.52	0.24	15.53	0.24
							MMRE	0.46	**MMRE**	0.47
							MdMRE	0.32	**MdMRE**	0.27
							PRED(30)	0.47	**PRED(30)**	0.53

Table A.3. Exploratory case study 2 detailed results - MicroM v1.0 (Int: Interaction, Com: Communication, Pro: Process, Eff-Ac: Actual effort in person-hour, Eff-Pr: Predicted Effort in person-hour, Lin. Reg: Linear Regression, Mult. Lin. Reg: Multiple Linear Regression).

Item	**MicroM Size**				**Eff-Ac**	**Lin. Reg.**		**Mult. Lin. Reg.**	
No	**Int**	**Com**	**Pro**	**Total**		**Eff-Pr**	**MRE**	**Eff-Pr**	**MRE**
1	2	1	0	3	2.50	4.20	0.68	4.06	0.62
2	2	1	0	3	2.50	4.20	0.68	4.06	0.62
3	4	2	0	6	4.00	6.54	0.63	6.40	0.60
4	1	1	1	3	4.00	4.20	0.05	3.94	0.02
5	3	2	0	5	4.50	5.76	0.28	5.34	0.19
6	2	1	0	3	6.00	4.20	0.30	4.06	0.32
7	4	5	0	9	6.5	8.87	0.35	7.06	0.09
8	2	1	3	6	7.00	6.54	0.07	6.87	0.02
9	6	3	2	11	8.00	10.43	0.30	10.62	0.33
10	5	1	1	7	8.00	7.31	0.09	8.18	0.02
11	2	1	3	6	10.00	6.54	0.35	6.87	0.31
12	6	2	8	16	11.50	14.33	0.25	16.02	0.39
13	8	6	4	18	13.00	15.89	0.22	15.27	0.17
14	5	2	5	12	15.00	11.21	0.25	12.15	0.19
15	16	9	0	25	18.00	21.35	0.19	20.67	0.15
16	8	4	4	16	20.00	14.33	0.28	14.83	0.26
17	10	6	1	17	20.50	15.11	0.26	14.58	0.29
						MMRE	0.31	**MMRE**	0.27
						MdMRE	0.28	**MdMRE**	0.26
						PRED(30)	0.71	**PRED(30)**	0.59

Table A.4. Evaluation case study 1 detailed results - COSMIC (Eff-Ac: Actual effort in person-hour, Eff-Pr: Predicted Effort in person-hour, Lin. Reg: Linear Regression, Mult. Lin. Reg: Multiple Linear Regression).

Item	COSMIC Size					Eff-Ac	Lin. Reg.		Mult. Lin. Reg.	
No	Entry	Read	Write	Exit	Total		Eff-Pr	MRE	Eff-Pr	MRE
1	0	2	0	2	4	6	5.13	0.15	6.10	0.02
2	3	3	0	3	9	10	9.62	0.04	9.96	0.00
3	1	1	0	1	3	7	4.23	0.40	4.31	0.38
4	1	1	2	1	5	5	6.03	0.21	5.91	0.18
5	2	0	0	2	4	3	5.13	0.71	4.79	0.60
6	2	0	0	2	4	3	5.13	0.71	4.79	0.60
7	1	0	1	1	3	5	4.23	0.15	3.94	0.21
8	1	1	0	3	5	5	6.03	0.21	6.58	0.32
9	2	0	0	2	4	4	5.13	0.28	4.79	0.20
10	1	1	0	2	4	4	5.13	0.28	5.44	0.36
11	1	1	0	1	3	4	4.23	0.06	4.31	0.08
12	1	1	0	1	3	4	4.23	0.06	4.31	0.08
13	2	0	0	3	5	5	6.03	0.21	5.92	0.18
14	2	2	1	3	8	6	8.72	0.45	9.07	0.51
15	2	0	0	3	5	9	6.03	0.33	5.92	0.34
16	1	2	0	2	5	4	6.03	0.51	6.61	0.65
17	1	1	0	1	3	3	4.23	0.41	4.31	0.44
18	1	1	0	1	3	4	4.23	0.06	4.31	0.08
19	1	1	0	1	3	5	4.23	0.15	4.31	0.14
20	2	0	0	2	4	5	5.13	0.03	4.79	0.04
21	1	1	1	3	6	5	6.93	0.39	7.38	0.48
22	1	1	0	1	3	6	4.23	0.30	4.31	0.28
23	1	2	0	1	4	8	5.13	0.36	5.48	0.31
24	1	1	0	1	3	3	4.23	0.41	4.31	0.44
25	1	1	0	1	3	3	4.23	0.41	4.31	0.44
26	1	1	1	1	4	8	5.13	0.36	5.11	0.36
27	2	1	0	2	5	4	6.03	0.51	5.96	0.49
28	3	0	0	3	6	8	6.93	0.13	6.44	0.19
29	1	1	0	3	5	15	6.03	0.60	6.58	0.56
30	1	1	1	1	4	6	5.13	0.15	5.11	0.15
31	3	0	0	3	6	3	6.93	1.31	6.44	1.15
32	1	1	1	1	4	7	5.13	0.27	5.11	0.27
33	2	0	2	1	5	6	6.03	0.00	5.26	0.12
34	1	1	0	1	3	5	4.23	0.15	4.31	0.14
35	1	0	1	2	4	4	5.13	0.28	5.07	0.27

Table A.4 (cont.). Evaluation case study 1 detailed results - COSMIC (Eff-Ac: Actual effort in person-hour, Eff-Pr: Predicted Effort in person-hour, Lin. Reg: Linear Regression, Mult. Lin. Reg: Multiple Linear Regression).

Item	**COSMIC Size**					**Eff-Ac**	**Lin. Reg.**		**Mult. Lin. Reg.**	
No	**Entry**	**Read**	**Write**	**Exit**	**Total**		**Eff-Pr**	**MRE**	**Eff-Pr**	**MRE**
36	1	0	2	4	7	7	7.83	0.12	8.14	0.16
37	1	1	0	1	3	4	4.23	0.06	4.31	0.08
38	4	0	0	4	8	12	8.72	0.27	8.09	0.33
39	2	0	0	3	5	4	6.03	0.51	5.92	0.48
40	2	0	2	4	8	12	8.72	0.27	8.66	0.28
41	2	0	2	4	8	7	8.72	0.25	8.66	0.24
42	2	0	0	2	4	5	5.13	0.03	4.79	0.04
43	2	0	0	2	4	3	5.13	0.71	4.79	0.60
44	1	0	1	1	3	3	4.23	0.41	3.94	0.31
							MMRE	0.31	**MMRE**	0.31
							MdMRE	0.28	**MdMRE**	0.28
							PRED(30)	0.59	**PRED(30)**	0.52

Table A.5. Evaluation case study 1 detailed results - MicroM (Fun: Functional Level Events, Arc: Architectural Level Events, Alg: Algorithmic Level Events, Eff-Ac: Actual effort in person-hour, Eff-Pr: Predicted Effort in person-hour, Lin. Reg: Linear Regression, Mult. Lin. Reg: Multiple Linear Regression).

Item	MicroM Size				Eff-Ac	Lin. Reg.		Mult. Lin. Reg.	
No	Fun	Arc	Alg	Total		Eff-Pr	MRE	Eff-Pr	MRE
1	2	2	0	4	6	5.20	0.13	5.09	0.15
2	1	3	4	8	10	9.24	0.08	9.67	0.03
3	1	2	2	5	7	6.21	0.11	6.48	0.07
4	1	3	3	7	5	8.23	0.65	8.45	0.69
5	0	2	0	2	3	3.18	0.06	2.98	0.01
6	0	2	0	2	3	3.18	0.06	2.98	0.01
7	1	1	1	3	5	4.19	0.16	4.51	0.10
8	1	4	2	7	5	8.23	0.65	7.97	0.59
9	0	3	2	5	4	6.21	0.55	6.17	0.54
10	0	2	2	4	4	5.20	0.30	5.42	0.36
11	0	2	0	2	4	3.18	0.21	2.98	0.26
12	0	2	1	3	4	4.19	0.05	4.20	0.05
13	0	3	2	5	5	6.21	0.24	6.17	0.23
14	2	3	2	7	6	8.23	0.37	8.28	0.38
15	1	5	3	9	9	10.25	0.14	9.94	0.10
16	1	1	1	3	4	4.19	0.05	4.51	0.13
17	1	1	0	2	3	3.18	0.06	3.29	0.10
18	1	1	0	2	4	3.18	0.21	3.29	0.18
19	1	2	0	3	5	4.19	0.16	4.03	0.19
20	0	4	0	4	5	5.20	0.04	4.46	0.11
21	2	4	2	8	5	9.24	0.85	9.02	0.80
22	1	2	1	4	6	5.20	0.13	5.25	0.12
23	1	2	1	4	8	5.20	0.35	5.25	0.34
24	1	1	0	2	3	3.18	0.06	3.29	0.10
25	2	1	0	3	3	4.19	0.40	4.34	0.45
26	3	1	1	5	8	6.21	0.22	6.62	0.17
27	1	2	0	3	4	4.19	0.05	4.03	0.01
28	0	3	1	4	8	5.20	0.35	4.94	0.38
29	1	4	4	9	15	10.25	0.32	10.42	0.31
30	2	2	1	5	6	6.21	0.04	6.31	0.05
31	0	3	0	3	3	4.19	0.40	3.72	0.24
32	2	2	1	5	7	6.21	0.11	6.31	0.10
33	1	2	1	4	6	5.20	0.13	5.25	0.12
34	1	2	1	4	5	5.20	0.04	5.25	0.05

Table A.5 (cont.). Evaluation case study 1 detailed results - MicroM (Fun: Functional Level Events, Arc: Architectural Level Events, Alg: Algorithmic Level Events, Eff-Ac: Actual effort in person-hour, Eff-Pr: Predicted Effort in person-hour, Lin. Reg: Linear Regression, Mult. Lin. Reg: Multiple Linear Regression).

Item	**MicroM Size**				**Eff-Ac**	**Lin. Reg.**		**Mult. Lin. Reg**	
No	**Fun**	**Arc**	**Alg**	**Total**		**Eff-Pr**	**MRE**	**Eff-Pr**	**MRE**
35	1	2	0	3	4	4.19	0.05	4.03	0.01
36	2	2	1	5	7	6.21	0.11	6.31	0.10
37	1	1	1	3	4	4.19	0.05	4.51	0.13
38	0	4	3	7	12	8.23	0.31	8.14	0.32
39	0	2	2	4	4	5.20	0.30	5.42	0.36
40	2	4	2	8	12	9.24	0.23	9.02	0.25
41	2	2	2	6	7	7.22	0.03	7.53	0.08
42	0	4	1	5	5	6.21	0.24	5.69	0.14
43	0	2	1	3	3	4.19	0.40	4.20	0.40
44	1	1	0	2	3	3.18	0.06	3.29	0.10
						MMRE	0.22	**MMRE**	0.21
						MdMRE	0.15	**MdMRE**	0.13
						PRED(30)	0.73	**PRED(30)**	0.70

Table A.6. Evaluation case study 2 detailed results - COSMIC (Eff-Ac: Actual effort in person-hour, Eff-Pr: Predicted Effort in person-hour, Lin. Reg: Linear Regression, Mult. Lin. Reg: Multiple Linear Regression).

Item	**COSMIC Size**					**Eff-Ac**	**Lin. Reg.**		**Mult. Lin. Reg.**	
No	**Entry**	**Read**	**Write**	**Exit**	**Total**		**Eff-Pr**	**MRE**	**Eff-Pr**	**MRE**
1	4	4	1	4	13	8.75	13.87	0.58	14.97	0.71
2	3	4	0	2	9	21	13.69	0.35	15.78	0.25
3	2	3	0	1	6	14	13.56	0.03	14.32	0.02
4	7	1	2	4	14	19	13.91	0.27	14.95	0.21
5	1	1	2	3	7	10.5	13.60	0.30	8.52	0.19
6	1	2	4	2	9	10.5	13.69	0.30	11.36	0.08
7	3	5	1	4	13	16	13.87	0.13	15.04	0.06
8	3	3	0	5	11	12.5	13.78	0.10	11.06	0.12
9	3	5	1	4	13	15	13.87	0.08	15.04	0.00
10	1	1	2	2	6	10.5	13.56	0.29	9.65	0.08
11	8	3	2	9	22	16	14.26	0.11	13.22	0.17
12	8	10	4	12	34	19	14.79	0.22	19.52	0.03
13	6	1	1	4	34	8.5	14.79	0.74	13.50	0.59
14	6	3	2	3	12	18	13.82	0.23	17.48	0.03
15	4	2	1	4	14	7	13.91	0.99	12.31	0.76
							MMRE	0.31	**MMRE**	0.22
							MdMRE	0.27	**MdMRE**	0.12
							PRED(30)	0.67	**PRED(30)**	0.80

Table A.7. Evaluation case study 2 detailed results - MicroM (Fun: Functional Level Events, Arc: Architectural Level Events, Alg: Algorithmic Level Events, Eff-Ac: Actual effort in person-hour, Eff-Pr: Predicted Effort in person-hour, Lin. Reg: Linear Regression, Mult. Lin. Reg: Multiple Linear Regression).

Item	**MicroM Size**				**Eff-Ac**	**Lin. Reg.**		**Mult. Lin. Reg.**	
No	**Fun**	**Arc**	**Alg**	**Total**		**Eff-Pr**	**MRE**	**Eff-Pr**	**MRE**
1	7	1	0	8	8.75	11.15	0.27	9.81	0.12
2	10	0	4	14	21	14.23	0.32	17.10	0.19
3	4	1	2	7	14	10.63	0.24	12.92	0.08
4	9	5	2	16	19	15.26	0.20	17.25	0.09
5	7	0	0	7	10.5	10.63	0.01	8.99	0.14
6	9	1	1	11	10.5	12.69	0.21	12.10	0.15
7	13	0	5	18	16	16.29	0.02	19.60	0.23
8	4	2	3	9	12.5	11.66	0.07	15.61	0.25
9	11	0	1	12	15	13.21	0.12	11.70	0.22
10	13	0	0	13	10.5	13.72	0.31	10.25	0.02
11	19	3	1	23	16	18.86	0.18	15.84	0.01
12	14	4	3	21	19	17.83	0.06	19.35	0.02
13	9	3	0	12	8.5	13.21	0.55	11.87	0.40
14	9	3	1	13	18	13.72	0.24	13.74	0.24
15	11	1	0	12	7	13.21	0.89	10.65	0.52
						MMRE	0.25	**MMRE**	0.18
						MdMRE	0.21	**MdMRE**	0.15
						PRED(30)	0.73	**PRED(30)**	0.87

Table A.8. Evaluation case study 3 detailed results - COSMIC (Eff-Ac: Actual effort in person day, Eff-Pr: Predicted Effort in person day, Lin. Reg: Linear Regression, Mult. Lin. Reg: Multiple Linear Regression).

Item	**COSMIC Size**					**Eff-Ac**	**Lin. Reg.**		**Mult. Lin. Reg.**	
No	**Entry**	**Read**	**Write**	**Exit**	**Total**		**Eff-Pr**	**MRE**	**Eff-Pr**	**MRE**
1	20	7	3	29	59	127	172.60	0.36	196.37	0.55
2	25	15	10	33	83	348	245.83	0.29	288.82	0.17
3	22	10	12	41	85	177	251.93	0.42	204.58	0.16
4	52	0	0	78	130	444	389.24	0.12	396.59	0.11
5	118	48	73	163	402	1217	1219.19	0.00	1219.11	0.00
6	17	2	0	26	45	103	129.88	0.26	142.44	0.38
7	5	10	0	16	31	112	87.16	0.22	68.18	0.39
8	17	3	14	20	54	205	157.34	0.23	174.18	0.15
9	8	19	1	27	55	163	160.39	0.02	116.65	0.28
10	27	18	5	47	97	207	288.55	0.39	282.44	0.36
11	8	0	0	12	20	52.75	53.60	0.02	66.41	0.26
							MMRE	0.21	**MMRE**	0.26
							MdMRE	0.23	**MdMRE**	0.26
							PRED(30)	0.73	**PRED(30)**	0.64

Table A.9. Evaluation case study 3 detailed results - MicroM (Fun: Functional Level Events, Arc: Architectural Level Events, Alg: Algorithmic Level Events, Eff-Ac: Actual effort in person-day, Eff-Pr: Predicted Effort in person-day, Lin. Reg: Linear Regression, Mult. Lin. Reg: Multiple Linear Regression).

Item	**MicroM Size**				**Eff-Ac**	**Lin. Reg.**		**Mult. Lin. Reg.**	
No	**Fun**	**Arc**	**Alg**	**Total**		**Eff-Pr**	**MRE**	**Eff-Pr**	**MRE**
1	33	9	15	57	127	171.57	0.35	159.90	0.26
2	84	0	16	100	348	287.62	0.17	274.05	0.21
3	83	0	19	102	177	293.01	0.66	268.44	0.52
4	78	26	26	130	444	368.58	0.17	382.48	0.14
5	389	2	55	446	1217	1221.40	0.00	1221.04	0.00
6	24	13	10	47	103	144.58	0.40	156.26	0.52
7	16	0	7	23	112	79.81	0.29	65.50	0.42
8	54	0	3	57	205	171.57	0.16	189.38	0.08
9	31	0	7	38	163	120.29	0.26	113.12	0.31
10	55	12	10	77	207	225.54	0.09	249.45	0.21
11	12	4	4	20	52.75	71.71	0.36	76.10	0.44
						MMRE	0.27	**MMRE**	0.28
						MdMRE	0.26	**MdMRE**	0.26
						PRED(30)	0.64	**PRED(30)**	0.55

VITA

Hüseyin Ünlü

Academic Experience

2016 – 2019	**Research/Teaching Assistant** *Department of Computer Engineering* *Middle East Technical University Northern Cyprus Campus*
2019 – 2024	**Research/Teaching Assistant** *Department of Computer Engineering* *Izmir Institute of Technology*

Education

2010 – 2016	**BSc in Computer Engineering** *Middle East Technical University Northern Cyprus Campus*
2016 – 2019	**MSc in Sustainable Environment and Energy Systems** *Middle East Technical University Northern Cyprus Campus*

Services

2022	**Organization Committee Member** *The Joint Conference of the 31st International Workshop on Software Measurement (IWSM) and the 16th International Conference on Software Process and Product Measurement (Mensura). Çeşme, İzmir.*
2023	**Proceedings Co-Chair** *The Joint Conference of the 32nd International Workshop on Software Measurement (IWSM) and the 17th International Conference on Software Process and Product Measurement (Mensura). Rome, Italy.*
2024	**Proceedings Co-Chair** *The Joint Conference of the 33rd International Workshop on Software Measurement (IWSM) and the 18th International Conference on Software Process and Product Measurement (Mensura). Montreal, Canada.*

Publications

Journal Articles Indexed by SCI and SSCI

2021	Ünlü, H, Yesilada, Y. Transcoding Web Pages via Stylesheets and Scripts for Saving Energy on the Client. *Software: Practice and Experience.*
2022	Ünlü, H, Bilgin, B, Demirors, O. A Survey on Organizational Choices for Microservice-based Software Architectures. *Turkish Journal of Electrical Engineering & Computer Sciences.*
2023	Ünlü, H, Yürüm, O.R, Özcan-Top, Ö, Demirors, O. How Software Practitioners Perceive Work-Related Barriers and Benefits Based on their Educational Background: Insights from a Survey Study. *IEEE Software.*
2024	Ünlü, H, Kennouche, D.E, Kilinc Soylu, G, Demirörs, O. Microservice-Based Projects in Agile World: A Structured Interview. *Information and Software Technology.*
2024	Ünlü, H, Garousi, V, Demirörs, O. Readiness and Maturity Models for Industry 4.0: A Systematic Literature Review. *Journal of Software: Evolution and Process.*
2024	Hacaloğlu, T, Ünlü, H, Yıldız, A, Demirörs, O. Software Size Measurement: Bridging Research and Practice. *IEEE Software.*
2024	Ünlü, H, Yürüm, O.R, Yıldız, A, Ö, Demirors, O. Application of a Size Measurement Standard in Data Warehouse Projects. *Software: Practice and Experience*.

Articles Published in Other Journals

2021	Ünlü, H, Bilgin, B, Demirörs, O. Türkiye'deki Yazılım Organizasyonlarının Mikroservis Tabanlı Mimaride Uyguladığı Analiz ve Tasarım Yöntemleri Üzerine Bir Araştırma. *EMO Bilimsel Dergi.*

Refereed Congress/Symposium Publications in Proceedings

2018	Ünlü, H, Yesilada, Y. Energy Efficient Mobile Web via Scripts&Stylesheets Based Transcoding. *12th Turkish National Software Engineering Symposium (UYMS).*
2019	Yaşar, D, Caner, S, Ünlü, H, Yıldız, A, Karabayır, A.K, Sokat, B, Bilgin, B, Arık, G, Anıl, L, Özen, O.S, Demirörs, O. Agility Assessment AgilityMod and Process Capability Assessment SPICE Models Field Applications. *13th Turkish National Software Engineering Symposium (UYMS).*
2020	Hacaloğlu, T, Ünlü, H, Demirörs, O, Abran, A. COSMIC light vs COSMIC Classic Manual: Case Studies in Functional Size Measurement. *The Joint Conference of the 30th International Workshop on Software Measurement (IWSM) and the 15th International Conference on Software Process and Product Measurement (Mensura).*
2020	Bilgin, B, Ünlü, H, Demirors, O. Analysis and Design of Microservices: Results from Turkey. *14th Turkish National Symposium on Software Engineering (UYMS).*
2021	Ünlü, H, Yalçın, A.G, Öztürk, D, Akkaya, G, Kalecik, M, Ekici, N.U, Orhan, O, Çiftçi, O, Yumlu, S, Demirörs, O. Software Effort Estimation Using ISBSG Dataset: Multiple Case Studies. *15th Turkish National Software Engineering Symposium (UYMS).*
2021	Ünlü, H, Hacaloğlu, T, Leblebici, O, Demirörs, O. Effort Prediction for Microservices: A Case Study. *15th Turkish National Software Engineering Symposium (UYMS).*
2021	Unlu, H, Tenekeci, S, Yıldız, A, Demirors, O. Event Oriented vs Object Oriented Analysis for Microservice Architecture: An Exploratory Case Study. *47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA).*
2022	Ünlü, H, Yıldız, A, Demirörs, O. Effort Prediction with Limited Data: A Case Study for Data Warehouse Projects. *48th Euromicro Conference Series on Software Engineering and Advanced Applications (SEAA).*

2022	Ünlü, H, Hacaloğlu, T, Büber, F, Berrak, K, Leblebici, O, Demirörs, O. Utilization of Three Software Size Measures for Effort Estimation in Agile World: A Case Study. *48th Euromicro Conference Series on Software Engineering and Advanced Applications (SEAA).*
2022	Hacaloğlu, T, Say, B, Ünlü, H, Küçükateş Ömüral, N, Demirörs, O. A survey on COSMIC students estimation challenge. *The Joint Conference of the 31st International Workshop on Software Measurement (IWSM) and the 16th International Conference on Software Process and Product Measurement (Mensura).*
2023	Ünlü, H, Hacaloğlu, T, Küçükateş Ömüral, N, Çalışkanel, N, Leblebici, O, Demirörs, O. An Exploratory Case Study on Effort Estimation in Microservices. *49th Euromicro Conference Series on Software Engineering and Advanced Applications (SEAA).*
2023	Kılınç Soylu, G, Ünlü, H, Shafique, I, Demirörs, O. Size Measurement and Effort Estimation in Microservice-based Projects: Results from Pakistan. *The Joint Conference of the 32nd International Workshop on Software Measurement (IWSM) and the 17th International Conference on Software Process and Product Measurement (Mensura).*
2023	Ünlü, H, Kılınç Soylu, G, Shafique, I, Demirörs, O. Analysis, Design, and Test in Microservice-based Projects: Results from Pakistan. *4th Agility with Microservices Programming Workshop (AMP)*
2024	Ünlü, H, Tenekeci, S, Çiftçi, S, Oral, İ.B, Atalay, T, Hacaloğlu, T, Musaoğlu, B, Demirörs, O. Predicting Software Functional Size Using Natural Language Processing: An Exploratory Case Study. *50th Euromicro Conference Series on Software Engineering and Advanced Applications (SEAA).*
2024	Yürüm, O.R, Ünlü, H, Demirörs, O. Towards the Construction of a Software Benchmarking Dataset via Systematic Literature Review. *50th Euromicro Conference Series on Software Engineering and Advanced Applications (SEAA).*

2024	Tenekeci, S, Ünlü, H, Dikenelli, E, Selçuk, U, Kılınç Soylu, G, Demirörs, O: Predicting Software Size and Effort from Code Using Natural Language Processing. *The Joint Conference of the 33rd International Workshop on Software Measurement (IWSM) and the 18th International Conference on Software Process and Product Measurement (Mensura).*