Thesis for The Degree of Doctor of Philosophy

An Empirical Investigation of Microservices-based Software Architectures from a Socio-Technical Perspective

Hamdy Michael Ayas

Department of Computer Science & Engineering
Chalmers University of Technology | University of Gothenburg
Gothenburg, Sweden, 2024

An Empirical Investigation of Microservices-based Software Architectures from a Socio-Technical Perspective

Hamdy Michael Ayas

ISBN 978-91-8103-032-7
Doktorsavhandlingar vid Chalmers tekniska högskola, Ny serie nr 5490 .
ISSN 0346-718X

Division of Interaction Design and Software Engineering
Department of Computer Science & Engineering
Chalmers University of Technology | University of Gothenburg
Gothenburg, Sweden

This thesis has been prepared using LaTeX.
Printed by Chalmers Digitaltryck,
Gothenburg, Sweden 2024.

*“And if you find her poor, Ithaka won’t have fooled you.
Wise as you will have become, so full of experience...”
- C. P. Cavafy, “The City”*

Abstract

Background: As software systems grow, software organizations turn towards Microservices-based Software Architectures (MSAs) seeking modularity, service orientation, and cloud-based software delivery. Microservices are a way of structuring software systems into loosely coupled pieces that: 1) are developed and operated independently, 2) communicate and integrate with each other to compose a system, and 3) each piece has its individual business domain and resources. However, migrating towards microservices entails a lot of complexity. The change that a microservices migration predisposes has a socio-technical nature with changes in the technology, the organization, and software engineers.

Objective: Therefore, this thesis aims to draw a holistic framework describing the change that microservices introduce. Change that is introduced to the software, the software developing organizations, and importantly, the human practitioners developing the software. Specifically, the goal is to empirically demonstrate the journey towards MSAs, along with the decision-making, the technical artifacts change, and the practitioners' roles, responsibilities, and skills in microservices.

Method: The methodologies of this thesis are based on inductive reasoning and use a combination of qualitative and quantitative methodologies. Grounded Theory and Grounded Theory-based analysis are used on interview data, textual data that engineers share in Q&A websites (i.e., StackOverflow), software repositories, and job-ads. In addition, a survey and automated analysis of StackOverflow data are also used.

Results: The main findings of this thesis are regarding the comprehensive perspective on microservices migrations that take place in multiple dimensions (business, technical, organizational), in multiple levels of abstraction (architecture and system), and in multiple modes of change (technical and systemic migrations). Specifically, 22 decisions and 53 solution outcomes are identified in detail. This work does not only approach migrations as a technical endeavor but also as an endeavor with a strong social and business aspect to it, covering the basic elements of socio-technical systems as defined in literature. Hence, the thesis presents taxonomies on the roles, responsibilities, and skills of microservices practitioners. Specifically, 3 roles of microservices practitioners are described in detail, along with 11 technical competences clusters, 5 families of responsibilities, 8 themes of soft-skills, and 11 themes of hard-skills.

Conclusion: Microservice migration projects entail an inherent complexity due to the different dimensions that the change takes place on, as well as the distributed nature of microservices. This work helps to decompose this complexity and carry a detailed understanding of microservices migrations to future attempts. In addition, this thesis paves the way for preparing engineers to work with MSAs.

Keywords

Microservices, Microservices migrations, Software architecture migrations, Grounded Theory, StackOverflow Mining

Acknowledgment

First, I would like to express my deep appreciation to my supervisors Philipp Leitner and Regina Hebig. Your input and insights have been invaluable in shaping the research of this thesis. Even more importantly, your coaching during these five years helped to develop my research acumen and I will be forever grateful for that. Next, I would like to thank my examiner Miroslaw Staron, for the great feedback and directions towards quality during my PhD journey.

I would also like to extend my appreciation towards the opponent and examination committee of this thesis, for joining me in the final stages of my PhD, via reading and discussing my work.

Additionally, I would like to express my gratitude to Agneta, Christian, Wolfgang, Nir, Clara and Jenny for their support in all PhD matters. It is a priviledge to have university ordinances that ensure a supportive environment.

Next, I would like to thank Francisco and Hartmut for the joyfull collaboration we had. Special thanks go to current and past office mates and friends, Joel, Linda, Razan, and Georgios, for the friendly and colorful working atmosphere. I also thank Krishna, Ranim, Bea, Babu, Ricardo, Habib, Cristy, Wardah, and all colleagues in the IDSE division for the nice time.

I am grateful to everyone I met during these 5 years, for making Gothenburg feel like a home. You are all special and the best companions one could wish for. A special place in my heart will always be occupied from my childhood friends, Kyriakos Pe., Kyriakos Pa., Dimitris, Paris, George, Andreas A., Andreas S., Gabriel, and the rest of the gang back in Cyprus. Thank you for all the fun summers and winters.

Most importantly I would like to thank my family: my grandparents - Vasiliki and Kyriakos, my parents - Mohammad and Eleni, my beloved siblings - Sandy and Malek as well as my beloved niece and nephew - Maria and Ali. Last but not least, I would like to thank my lovely partner and life companion Georgia, for always rooting for me. I am grateful that you are by my side when powering through all kinds of waves, slopes, hills, mountains and cities in all kinds of sunshine, rain, snow, winds and icy cold. You are all my roots, my wings and my light.

I would like to acknowledge the financial support that this research has received from Vinnova (Sweden's innovation agency) under the grant number 2018-05010. Additionally, I wish to thank all the anonymous reviewers that provided constructive feedback to the research included in this thesis.

List of Publications

Appended publications

This thesis is based on the following publications:

[A] H. Michael Ayas, P. Leitner, R. Hebig "Facing the Giant: a Grounded Theory Study of Decision-Making in Microservices Migrations"
International Conference on Empirical Software Engineering and Measurement (ESEM2021), 2021.

[B] H. Michael Ayas, P. Leitner, R. Hebig "The Migration Journey Towards Microservices"
International Conference on Product-Focused Software Process Improvement (PROFES2021) 20(35), 2021.

[C] H. Michael Ayas, P. Leitner, R. Hebig "An Empirical Study of the Systemic and Technical Migration Towards Microservices"
Empirical Software Engineering (EMSE), 28, 85 (2023).

[D] H. Michael Ayas, P. Leitner, R. Hebig "The perceived impact and sequence of activities when transitioning to microservices"
IEEE International Conference on Service-Oriented System Engineering (SOSE), 2023.

[E] H. Michael Ayas, H. Fischer, P. Leitner, F.G. de Oliveira Neto "An Empirical Analysis of Microservices Systems Using Consumer-Driven Contract Testing"
48th Euromicro Conference Series on Software Engineering and Advanced Applications (SEAA), 2022

[F] H. Michael Ayas, R. Hebig, P. Leitner "An empirical investigation on the competences and roles of practitioners in Microservices-based Architectures"
To appear in The Journal of Systems & Software (JSS), 2024.

[G] H. Michael Ayas, R. Hebig, P. Leitner "The Roles, Responsibilities, and Skills of Engineers in the Era of Microservices-Based Architectures"
International Conference on Cooperative and Human Aspects of Software Engineering (CHASE 2024).

Other publications

The following publications were published during my PhD studies, or are currently in submission/under revision. However, they are not appended to this thesis, due to contents overlapping that of appended publications or contents not related to the thesis.

[a] M. Mortada, H. Michael Ayas, R. Hebig "Why do software teams deviate from scrum? reasons and implications"
International Conference on Software and Systems Processes (ICSSP2020), 2020.

Research Contribution

I (Hamdy Michael Ayas) was the main driver and contributor of papers A, B, C, D, F, G as they are appended in this thesis (all appended papers except Paper E). In Paper E I was the main driver, but contributed equally with the second author. A summary of the contributions is presented in Table 1, based on the Contributor Roles Taxonomy (CreditT) [1].

For Papers A, B, C, D, F, and G I was the main contributor in most categories of the taxonomy, as shown in Table 1. Specifically, I significantly contributed in the *Conceptualization, Data Curation, Formal Analysis, Investigation, Methodology, Software (when applicable), Validation, Visualization and Writing of the original draft.* I also facilitated the ways that co-authors (e.g., main and co-supervisor) contributed in the *Formal Analysis* (e.g., via data analysis guides), to ensure enough rigor in the chosen (qualitative) research methodologies. In addition, I facilitated the inclusion of co-authors' expertise in the *Conceptualization* of the topics discussed in this thesis, the *Methodology* and some *Data curation* activities.

For Paper E, I contributed with the *Conseptualization, Formal Analysis, Investigation, Validation, Methodology, Software, Supervision, Visualization, Writing of the original draft and Writing through review and editing.* The work of Paper E is based on a BSc thesis that I supervised, in which a partial re-analysis of the existing data gathered and a partial re-writing of the publication took place.

Role	**Paper A**	**Paper B**	**Paper C**	**Paper D**	**Paper E**	**Paper F**	**Paper G**
Conceptualization	✓	✓	✓	✓	✓	✓	✓
Data Curation	✓	✓	✓	✓		✓	✓
Formal Analysis	✓	✓	✓	✓	✓	✓	✓
Funding acquisition							
Investigation	✓	✓	✓	✓	✓	✓	✓
Methodology	✓	✓	✓	✓	✓	✓	✓
Project administration							
Resources							
Software			✓		✓	✓	✓
Supervision					✓		
Validation	✓	✓	✓	✓	✓	✓	✓
Visualization	✓	✓	✓	✓	✓	✓	✓
Writing - original draft	✓	✓	✓	✓	✓	✓	✓
Writing - review & editing					✓		

Table 1: The author's individual contributions to the appended papers of the thesis.

[1] https://casrai.org/credit/

Contents

Chapter 1

Introduction

As software becomes an integral part of organizations' value delivery, software applications grow in size, number of features, development teams, services delivered, and complexity. Software organizations turn towards Microservices-based Software Architectures (MSAs), as they seek modularity, service orientation, and interconnected cloud-based software delivery [1]. MSAs is a paradigm of structuring systems and their development at a large scale, complementing agile methodologies of software development [2]. Specifically, MSAs facilitate software development agility, by scoping the work of DevOps teams in the boundaries of independent and self-sufficient microservices [3]. One of the central ideas of MSAs is that of disentangling the delivery of an entire software system, from the delivery of any of its individual components [4]. Hence, organizations in many industries are increasingly adopting microservices technologies to structure their software [1,5].

Microservices are a way of structuring software systems into loosely coupled pieces of software that: 1) are developed and operated independently, 2) communicate and integrate with each other to compose a complete system, and 3) each piece has its own individual business domain and resources. Microservices are an incarnation of Service-Oriented Architectures (SOA) [2] and by adopting microservices, organizations can manage the complexity of their software [6] and deploy individual, vertically cut pieces of software autonomously and independently [7,8]. One of the principal premises of engineering software using microservices is that the process of engineering is centered around Domain-Driven Design and Development (DDD), which means that business function and business value delivery are at the core of the engineering process [4].

However, implementing MSAs is not an easy task and there are several technical challenges along the way [9]. Microservices are not necessarily suitable for every type of system, or they are not necessarily the solution to existing problems that systems face. There are cases where the use of MSAs had negative effects and were reverted (e.g., back to a monolith) [10]. In addition, MSAs can be used in new software systems, but very often, an existing system needs to be migrated to an MSA [6] and there is plenty that we do not know, specifically about migrations towards microservices. Hence, there is a need to study the process of migrating and essentially re-engineering the architecture of software [11].

However, re-architecting is a complicated, time-consuming, and often messy process, with a lot of technical change required entailing several potential technical choices. There are more aspects (than just the source code) that change during migrations, such as integration and deployment methods, testing, and many more [12–15]. In addition, migration processes are often taking years to complete, making it hard to keep track of the process. Migrations are often unstructured and take place in ad-hoc manners (they are not systematic or methodical, but rather take place on a trial and error basis) [16]. They are also not well known since they are not always systematically recorded [6]. There is knowledge from practice and academia on how to technically enact MSA migrations, but many migrations are rarely aggregated to show engineers a more generic view of change, and thus, engineers often learn along the way.

Furthermore, re-architecting is also complex, since the technical changes are only one part of the undertaken change in migrations, accompanied by social changes and decisions that are not only technical but rather human-centric. For example, applying automated software decomposition tools on source code [17] does not mean that the system is migrated to an MSA. Architectural migrations are heavy in decision-making [18], either at an individual-level, team-level, or organizational-level (i.e., company-wide) [19]. Migrations predispose changes to the profiles (technical or non-technical) of practitioners and engineers involved, with new demands and topics of operation that need to be accounted for [12]. Hence, the evolution of the structures that a microservices migration predisposes has a socio-technical nature, comprising a technical, an organizational, and a social aspect [20]. The socio-technical concerns of such migrations matter at all levels of abstraction in a system (e.g., classes, modules, services, etc.) and influence each other. On the one hand, there are small, simple development details that can influence grand design choices. On the other hand, there are grand design choices that can influence development in small, simple development details.

This thesis is unraveling the way towards MSAs and draws a holistic framework of understanding, about the change that microservices introduce. Change that is introduced to the software, the software developing organizations, and very importantly, the human practitioners developing the software. Consequently, this thesis entails research that is grounded on these three dimensions. Specifically, the goal is to empirically demonstrate the journey towards microservices, along with the decision-making, the technical artifacts change, and the practitioners' roles, responsibilities, and skills in microservices. Ultimately, the thesis aims to establish microservices migration processes that balance between technological change, organizational change and at the same time maintain human attributes as a core characteristic of migration processes. The thesis is structured in eight chapters from which this introductory chapter is the synopsis of the thesis. In Section 1.1 the theoretical framework and related work are discussed, describing the foundations of this work and putting the thesis in the context of related research. Section 1.2 describes the scope of the thesis, providing the objectives and Research Questions (RQs) addressed in the thesis. Section 1.3 includes a detailed overview of the Research methodologies, including a summary of the data gathering, processing, and analysis conducted. Section 1.4 presents the contributions of each individual appended paper, along with answers to the RQs. Section 1.5 contains the Discussion, with implications of this thesis' results, discussions on how the results address the research objectives, as well as the results' relevance to practice and research community. Finally, Section 1.6 draws the Conclusions of the thesis. In the remaining of the thesis, chapters 2 - 8 correspond to the appended publications of the thesis (Papers A - G).

1.1 Theoretical Framework & Related Work

Microservices are a way of structuring systems into loosely coupled pieces that are developed and operated independently, each with its own individual domains and resources. These individual pieces communicate with each other to compose a complete system through decentralized continuous delivery [6]. A system based on microservices is therefore composed as a set of small services, individually running in their own process, and communicating with lightweight mechanisms [4]. Hence, microservices have fine-grained interfaces (e.g., API endpoints) of independently deployable services [6, 21].

MSAs have certain characteristics. In principle, microservices are often small individual pieces of functionality and therefore they are often deployed in containers or even as functions-as-a-service [22]. Furthermore, business-driven development practices are critical to accompany agile practices that are often adopted by software development teams [2, 4]. In addition, microservices follow cloud-native application design principles [6]. Such principles allow them to be technology-independent and thus, have a polyglot nature (i.e., each microservice can be written in its own programming language). Additionally, microservices are characterized by their persistence strategies, especially in managing their own state [4] and having (in principle) their own database [23].

Often MSAs are described as an incarnation of Service Oriented Architectures (SOA) that aim to address the need for more flexible, loosely coupled compositions of services [21]. The difference between microservices and SOA is not necessarily in their architectural style but rather in the implementation of the architecture. Specifically, microservices embrace, leverage, and add on principles and patterns from SOA (loose coupling, service contracts, etc.) [2].

In microservices, Domain Driven Decomposition/Design (DDD) is key [4]. DDD is the business-centric or feature-centric design and development of software. In addition, microservices are about separating functionality based on different criteria rather than just functional concerns (horizontal splitting of backend, frontend, and data layers). For example, microservices can be structured in a system based on the number of features that are provided to the end users, the number of developers, and the number of users that use parts of a software system (vertical splitting) [22]. This way a MSA can facilitate organizations to achieve improved scalability, maintainability, and reduced time to market [4, 9].

This section discusses related work on the topics that this thesis is developed on. Related work on microservices migrations is followed by summarizing seminal work on human-aspects in software engineering.

1.1.1 Benefits of Microservices

The basic architecture of software contains software implementing the applications' logic (e.g., backend), potentially a data source (e.g., database, sensors, etc.), and an interface of interactions with consumers of the software (e.g., frontend, information exports, etc.). As software requirements grow, software evolves and grows with new features as well as modifications and maintenance of existing features. In fact, software often grows in size and complexity so much, that value delivery through software becomes slow and costly. That is because systems are often fixated on sub-optimal (seemingly) irreversible design choices, or accumulate technical debt making them unmaintainable [24]. However, continuous value delivery to customers is important, and technical debt needs to be repaid eventually to evolve software systems. Organizations need software systems to maintain their capacity to change fast with the development of new features, updating existing features, and combining features to deliver new digital services [19]. Consequently, as software systems evolve and scale faster than their underlying structures, software architecture transitions to modern, cutting-edge paradigms of development are becoming more and more common [25, 26].

Releasing new features and maintaining existing software becomes challenging as software systems grow large. Technical dept gets accumulated that binds organizations to design choices that have points of no return and digital services that cannot be modernized easily or fast [27]. This difficulty of modernizing digital services is a challenge of organizational agility [2] and can also lead to obsolescence in products that are offered by organizations [28]. Hence, software systems need to be structured more meticulously and MSAs are a way to do so [9].

Microservices have many advantages. First, microservices are about focusing on one thing and doing it well [9]. Also, microservices enable faster releases of functionality and independent scaling and maintenance [6, 27]. The polyglot nature of MSAs enables technology diversity and faster adoption of different technologies, depending on the requirements at hand [2]. Additionally, an MSA enables the separation of parts with high-security requirements design and allows multiple points of failure to achieve resilience [29]. Some of the effects of microservices are a response to business change, enabling different workloads for cost improvements, higher value delivery, organizational agility, and decentralized governance [13].

1.1.2 Microservices Migrations

As software systems grow large, both in size and complexity, it becomes difficult to update them and thus, they need to be re-structured [30]. That is because software systems often need to change fast with the development of new features, updating of existing ones, and combining/aggregating applications or digital services [11]. Such a re-structuring includes the evolution of monoliths to Service-Oriented-Architectures (SOA) and even further to microservices [2]. Monolithic software systems are tightly coupled pieces of software that have challenges in maintaining them effectively and updating them fast. SOAs are a step towards organizing software with separation of concerns, functional and non-functional requirements, and providing information as services [31]. MSAs go a step further and they are a way of structuring systems into loosely coupled pieces that are developed and operated independently, each with its domains and resources [2]. These individual pieces communicate with each other to compose a complete system [4]. Existing research investigates how to use business logic, domain, and existing solutions [13, 16], but there is further room for engineering requirements that are specific to microservices migrations.

Migration projects are not simple, since migrating a system towards microservices (e.g., from a monolithic architecture) is a long endeavor with many things to consider and an inherent complexity [13]. Hence, there is a need to empirically investigate the details of migrations comprehensively from different points of view [30]. Existing formal models can give guidance on how to track and split technical artifacts of the system [18, 32]. However, there are not many empirical investigations on the process of designing microservices-based architectures. Empirical evidence on migration projects can bring light to such practices as well as prepare practitioners for the expected migration journey and what activities such a journey entails [33]. Hence, studying and understanding how companies make their transitions towards MSAs can also provide a detailed theoretical basis to researchers on the different aspects of migrations [29]. Research and best practices stemming from industry provide some approaches on migrations, covering many aspects [13, 23]. There are different ways to split the software and transform a system into microservices technically. Figure 1.1 showcases the landscape on which such approaches exist.

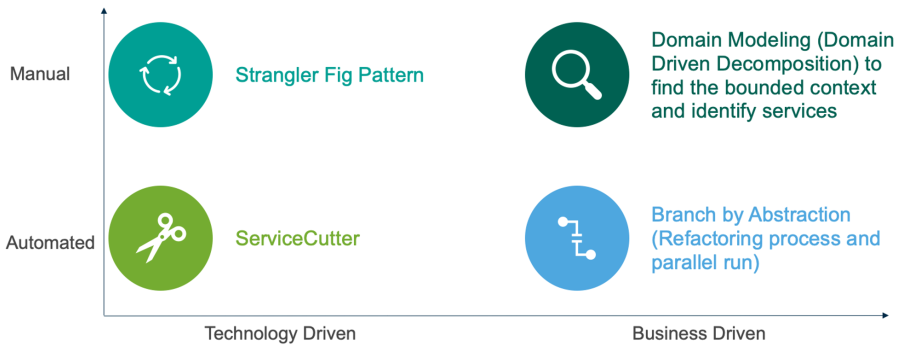
Manual
Automated
Strangler Fig Pattern
Domain Modeling (Domain Driven Decomposition) to find the bounded context and identify services
ServiceCutter
Branch by Abstraction (Refactoring process and parallel run)
Technology Driven
Business Driven

Figure 1.1: Categorization of microservices decomposition approaches

There are on the one hand manual approaches that deal with isolating through manual code analysis specific parts of the software to deconstruct services [13]. Such approaches have a high awareness of context and include extensive human input. Manual approaches that are driven by the business side adopt loyally Domain Driven Design (DDD) in the decomposition process, meaning that the bounded contexts between services are determined based on business function. For example, the business scope of a specific feature is driving the isolation of its service from the rest of the system. In repetition, this results in a solely business-driven split. Moreover, manual approaches that are driven by technology are more specific on the technologies used to decompose the software. For example, the Strangler Fig Pattern is about setting up an interface around the monolith and decomposing it by extracting one independent service at a time, that communicates with the monolith's interface.

On the other hand, there are automated approaches that take the source code and indicate potential splits. Specifically, there is static code analysis which is splitting by analyzing source code like class dependencies [34]. Another way is meta-data aided, which is through analyzing more abstract input data like UML, Use Cases, interfaces, commits, etc. [35]. Also, microservices can be split using workload-data-aided approaches [36]. Such approaches analyze measurements of operational data on a module on function level to define granularity. Furthermore, there are Dynamic Microservices Decomposition approaches. They are permanently changing services based on workload for example or other dimensions to re-calculate the best-fitting decomposition.

1.1.3 Human Aspects in Software Architecture

Engineering software entails substantial decision-making. Hence, considering human aspects and behaviors at the core of software engineering can help in enabling the full utilization of engineers' creative and mental capacity [37]. Currently, decisions for architectural designs, technical components, and development processes are highly intuitive and based on previous experiences of engineers [38]. The implications of decision-making are extensively studied in many other fields but moderately studied in software engineering [39]. In addition, decision-making is especially challenging when it takes place in groups since it is not common to have structured decision-making in groups [40]. Specifically, a central challenge of decision-making is to have well-structured problems that are well-defined beforehand [41]. Then, according to Zannier et al. (2007), it is more likely that engineers have a better understanding of the constituent elements of a problem and more likely to organize a rational process for addressing the problems at hand. Therefore, as decisions impact the evolution of the underlying structures of software, there is value in making decision-making processes explicit and transparent.

Moreover, MSAs call for a different tech stack than other architectures to deliver and operate software applications. For example, the tech stack of MSAs includes several auxiliary artifacts about testing, logging, and monitoring [12]. Therefore, MSAs are generating a wave of change in the development mindset of practitioners. Consequently, starting to define the specific roles and competences of engineers in them can be beneficial. Especially since the increased cognitive load put on microservices architects as well as the increased effort of designing, testing, and maintaining software in MSA is reported [2].

Fields that predispose a paradigm shift in development are re-defining the key roles and responsibilities of engineers [42]. Similarly, existing research has been investigating the characteristics of software engineering profiles, for example through programming language affinity [43] or personal traits [44]. Different ways to investigate and derive different technical roles of practitioners can be from their contributions in GitHub [45] or the skills that organizations are looking for in engineers during recruitement [46]. Moreover, existing research identifies detailed descriptions of the skills of practitioners in different software engineering sub-domains [47–50], but less so in microservices. Hence, there is a need to investigate further the skills of practitioners working with MSAs.

1.1.4 Summary of Research Gap

Existing research focuses largely on isolated technical aspects. Moreover, existing solutions (e.g., through program decomposition) often do not address the decision-making of engineers in migrations or the change of auxiliary technical artifacts (e.g., testing artifacts). Hence, this can lead to ad-hoc, disconnected processes from the rest of the organization or intended practices. For example, testing in microservices entails more complexity than testing in other architectures and thus, it is challenging to conceptualize how testing architecture changes with the introduction of MSAs.

Migrations entail decisions for bigger changes than just a systems' upgrades. They are transformative on organizations as a whole [3], and little research complements the technical aspects of microservices with the crucial non-technical aspects of microservices migrations. Hence, there is a lack of approaches providing details on the operational choices that software development teams and organizations make in migrations [12].

Even more importantly, there is a lack of empirical evidence on the practitioners' perceptions, competences, and skills. Specifically, even though there is an emergence of empirical studies capturing the perceptions of practitioners, there are calls for more empirical evidence on how practitioners experience microservices migrations. In addition, the tech stack of MSAs is dense and diverse and often new for practitioners. Capturing and organizing all the different types of technical competences and skills is challenging, but essential to scope the development of engineers with the required technical demands. Moreover, the responsibilities and soft-skills of microservices practitioners are under-investigated and they are equally important with the technical skills.

1.2 Research Scope

The main objective of this thesis is to bring structure to the often time-consuming, and challenging process of changing the software architecture towards microservices. In doing so, the aim is to maintain in the process the human aspects, since people are an integral part of shaping MSAs. Therefore, the goal is to empirically define the journey toward microservices, in light of the technological change, the organizational change, and the human attributes that comprise the core of migration processes.

To address the challenges described and the research gap, this thesis has 5 general research objectives, that are further broken down into 7 Research Questions.

1.2.1 Research Objectives

The following research objectives are covered in this thesis.

Research Objective 1 (Obj-1): Understand the decision-making process of migrations towards MSAs Migrating the software architecture towards microservices entails several decisions that influence the resulting change. Such decisions include not only technical choices but also choices regarding how to re-engineer the software system at hand. Obj-1 aims to provide a holistic understanding of the decisions that microservices migrations entail. Therefore, the goal is to present the process in which decisions are made by engineers to migrate their software architecture.

Research Objective 2 (Obj-2): Chart the migration journey towards MSAs Migrations towards MSAs are long endeavors that can entail many different activities. Obj-2 aims to chart the different activities of organizations that change their software architecture to microservices. Therefore, the goal is to decompose the migration journey into the different phases and activities that exist in such a journey, along with the potential modes of change that migrating organizations go through.

Research Objective 3 (Obj-3): Solution outcomes and changes in testing artifacts during microservices migrations During the journey towards MSAs, engineers end up with different solutions that implement their new architecture. In addition, testing in MSAs differs from previous architectures. Obj-3 aims to first aggregate (technical) solution outcomes of engineers in MSAs. In addition, Obj-3 targets to capture changes that appear in software artifacts, specifically testing artifacts. Therefore, the goal is to provide a set of solution outcomes from the work of practitioners, as well as the testing architecture that microservices have.

Research Objective 4 (Obj-4): Understand the organizational aspects of microservices migrations Migrations to MSAs are not only comprised of technical changes that take place but also changes that alter the enterprise in which the software operates. Therefore, Obj-4 aims to demonstrate the organizational change that comes with microservices and showcase the impact of MSAs on how organizations operate.

Research Objective 5 (Obj-5): Understand the human aspects of MSAs Microservices are designed and developed first and foremost by human engineers. Therefore, the human dimension of MSAs cannot be omitted, and the characteristics of microservices practitioners are investigated along the technical and organizational characteristics of MSAs. Obj-5 aims to gain an in-depth understanding of the roles, competences, responsibilities, technical skills, and interpersonal skills of engineers in MSAs. The goal is to complement the body of research on microservices with empirical evidence on human-aspects, to firstly understand the practitioners and secondly support them in their journey towards mastering microservices.

1.2.2 Research Questions

To achieve these objectives, a set of Research Questions (RQs) are presented below.

RQ1: *What is the decision-making process of organizations during a migration towards microservices?*

> This research direction empirically showcases decision-making and the typical options that organizations can choose in the decisions of migrations, addressing Obj-1. Hence, RQ1 empirically derives the decisions that organizations make during a migration towards microservices and when these decisions are made.

RQ2: *What is the migration journey that companies go through when transitioning towards microservices?*

> This research question aims to address the gap in the empirical understanding of migrations from the engineers' perspective, addressing Obj-2. RQ2 investigates what different levels can the migration journey take place, how these different levels are structured, and what is the sequence of basic migration activities.

RQ3: *What are the constituent elements of migration journeys that companies go through when transitioning towards microservices (from high level patterns to detailed solutions)?*

> This research question addresses Obj-2 and Obj-3, by pinpointing in detail the different activities that take place at different levels of abstraction across the organization. Specifically, activities and common solutions are identified on the technical and systemic level of migrations.

RQ4: *What is the architecture of testing artifacts in microservices?*

> RQ4 demonstrates the testing architecture of software systems based on microservices. In addition, this research direction provides empirical evidence on how microservices testing can differ from conventional testing best practices, addressing Obj-3.

RQ5: *What is the impact of MSAs on the organizational structures of migrating enterprises?*

This research question unravels with empirical evidence the change that happens in an organization that undertakes a microservices migration, addressing Obj-4. Hence, the impact of MSAs on the enterprise is captured and presented.

RQ6: *What are the roles and responsibilities of practitioners working with microservices?*

RQ6 taps into the identification of the typical technical roles that microservices practitioners have, along with their responsibilities. The research direction of this RQ is a first step towards empirically understanding the human-aspect in MSAs and addressing Obj-5.

RQ7: *What are the technical competences, hard-skills, and sof-skills of microservices practitioners?*

RQ7 organizes the competences and skills that microservices engineers have. Specifically, this RQ addresses Obj-5 by demonstrating both the technical skills that MSAs require, as well as the interpersonal and soft skills that are introduced with MSAs.

1.3 Research Methodologies

In this research, a combination of qualitative and quantitative methodologies have been used. The epistemological philosophy of the conducted research is predominantly based on inductive reasoning. Specifically, observations of the world are collected in the form of interviews, StackOverflow discussions and user tags, a survey, and job-ads posted online. The data are analyzed to recognize and identify patterns in microservices migrations and MSAs, drawing general conclusions from those patterns, and developing theories based on those patterns. As shown in Figure 1.2, every RQ is answered with at least one methodology, from at least one of the appended papers. The methodologies used are mainly Grounded Theory (GT) and thematic analysis using techniques from GT as described in literature on using GT in software engineering [51]. Additionally, a survey is conducted, along with a quantitative analysis of StackOverflow profile tags, and job-ads.

1.3.1 Interviews

The step of interview conducting and analysis is predominantly based on GT. During the interviews, the aim was to capture descriptions and contextual information about the overall perception and experiences of the interviewees' microservices migration journeys, capturing specifically the decisions during microservices migrations. Specifically, a GT study (the constructivist variance of GT, as defined by Stol et al. (2016)) with interviews was conducted for paper A [51]. The starting point was an initial research question that evolved throughout the development of paper A, as suggested in the literature for conducting studies based on constructivist GT [52]. The initial research

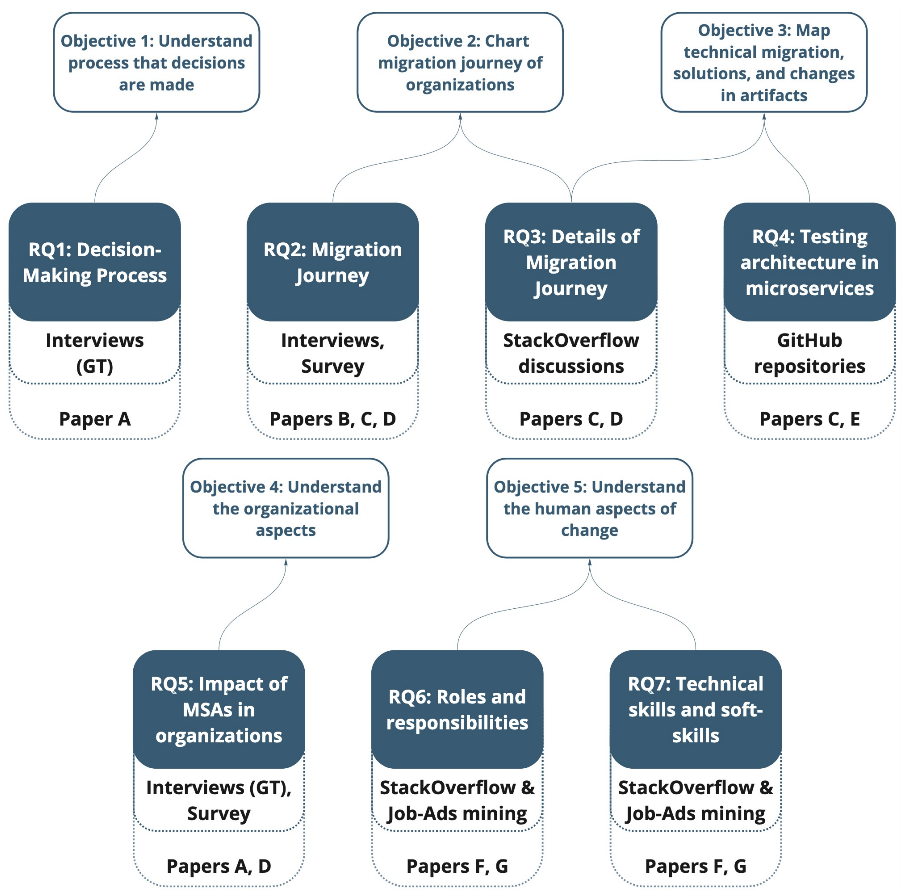
Objective 1: Understand process that decisions are made
Objective 2: Chart migration journey of organizations
Objective 3: Map technical migration, solutions, and changes in artifacts
RQ1: Decision-Making Process
Interviews (GT)
Paper A
RQ2: Migration Journey
Interviews, Survey
Papers B, C, D
RQ3: Details of Migration Journey
StackOverflow discussions
Papers C, D
RQ4: Testing architecture in microservices
GitHub repositories
Papers C, E
Objective 4: Understand the organizational aspects
Objective 5: Understand the human aspects of change
RQ5: Impact of MSAs in organizations
Interviews (GT), Survey
Papers A, D
RQ6: Roles and responsibilities
StackOverflow & Job-Ads mining
Papers F, G
RQ7: Technical skills and soft-skills
StackOverflow & Job-Ads mining
Papers F, G

Figure 1.2: Overview of Research Objectives, Research Questions, methodologies and corresponding papers

question was further specified and broken down into what could be addressed based on the data analysis of paper A. A semi-structured interview guide was used to conduct the interviews, which we constructed based on the initial research questions. However, participants were given significant freedom in describing their migration experiences. The data collection included interviews with 19 participants, from 16 organizations operating in different industries. Furthermore, an additional qualitative analysis took place on the same interview-based dataset. The same interviewing data gathering step provided data for the thematic analysis of paper B, where the migration journey towards microservices is empirically derived. The derived journey of paper B was provided as input to paper C.

The data processing of papers A, B, and C is transcribing each interview and formatting the transcripts for analysis (i.e., formatting in documents suitable for coding and keeping a record of codes). In the case of papers B and C, a pre-processing step took place to include data that were excluded in paper A. Specifically, a manual selection took place, excluding the parts of interviews that had to do with decision-making, keeping only the parts that described the migration journey.

The analysis of paper A includes initial coding where all the transcripts are analyzed, and interesting bits are highlighted. Then, focused coding is conducted where decision-making-related information is selected. Finally, theoretical coding aggregates and organizes all the captured information into a decision-making process. Furthermore, an additional qualitative analysis took place on the same interview-based dataset. The additional analysis step led to the results of paper B and the input or starting point of paper C. Specifically, the focused coding of paper A excluded a lot of information about the overall migration journey (since it was focused on the decision-making process). Hence, papers B and C capture the information on the overall migration journey of the interviewees and their microservices migrating organizations. Then thematic analysis techniques are used to draw a theory of the overall migration journey.

1.3.2 Survey

Moreover, in paper D a survey is conducted, gathering the perspectives of practitioners that worked with microservices. The survey's closed questions complement the open-ended questions of the previous methodology steps. 54 different engineers responded to the survey, revealing their perceptions on the impact of microservices as well as the sequence of microservices migrations activities. The raw survey responses were collected and formatted in a data format (CSV) to enable their analysis. The analysis of the survey for paper D captured the perceived impact and the sequence of activities of microservices migrations. The survey included questions about a set of impact areas and microservices migration activities, asking respondents to sequence them.

1.3.3 StackOverflow Discussions

This part of the methodology is a purely manual analysis of posts mined from StackOverflow, again using techniques from GT, as described in literature [51, 52]. StackOverflow is often the place that software engineers turn to, for sharing issues and challenges they face, along with potential solutions to their technical work issues [53]. Therefore, discussions arise in this Q&A forum containing a lot of details regarding engineers' concerns. Developers use posts from such websites to gather information, get ideas for solutions, and discuss their design decisions to validate them [54]. More importantly, software engineers share issues and challenges they face [55], along with potential solutions to their particular technical issues. The content that is shared among developers often includes information on the ways that they work, think, and tackle different issues [56].

This thesis derives from (sometimes lengthy) discussions of software engineers in StackOverflow detailed solutions in migration activities towards microservices. The data collection takes place by querying StackExchange, to gather questions that engineers posted as well as answers to those questions. Specifically, paper C makes use of StackExchange Data Explorer (SEDE), to gather discussions related to microservices migrations. Therefore, the queries in SEDE gather questions that have combinations of keywords such as *"microservices"* with keywords such as *"monolith"*, *"migration"*, and *"transition"*. These questions are filtered to include those that are discussed by at least one more peer engineer and have a positive total voting score. Then, the discussions of these questions are gathered and all the data are analyzed in detail manually. The analysis of the 215 gathered posts is qualitative and based on GT techniques. The purpose of this analysis is twofold. On the one hand, to evaluate the already developed theory from paper B and on the other hand, to extend the theory with detailed solutions as presented in paper C.

1.3.4 Mining StackOverflow Tags of User Profiles

In paper F an automated Python script gathers the tags associated with StackOverflow user profiles. Specifically, the script uses the StackExchange API to first gather the profiles of users who had posts about microservices transitions. Then, the script gathers the tags that are associated with those profiles. Paper F includes a fairly elaborate processing of the user-tags gathered. Specifically, inclusion and exclusion criteria are applied to achieve a trade-off between the feasibility of analysis, data completeness, and contextually sense-making results. First, the tags co-occurrence matrix is derived, indicating every pair of tags that appear together.

Paper F starts the analysis by applying the Louvain clustering method, decomposing the co-occurrence matrix into competences clusters. Subsequently, a manual analysis step is conducted on the derived clusters, capturing the themes of clusters and collections of clusters. Hence, the topics and technical areas that practitioners have competences in are identified. Finally, an analysis step is calculating the association score of user-profiles with the derived competences clusters, indicating the roles that microservices engineers have across different competences.

1.3.5 Analyzing Open Source Microservices Systems

Paper E mines GitHub to gather repositories that have a relatively mature microservices-based architecture. Specifically, the focus of the search in paper E is on software systems that use specific technologies, namely frameworks for Consumer-Driven Contract testing (CDC). Hence, the search identified technology traces of Pact and Spring Cloud Contracts (SCC) in repositories. Thereafter, the gathered repositories were filtered to include as close to real-world software as possible, removing software applications that were not authored by an organization. Consequently, the analysis of paper E includes 16 repositories of 22 microservices that compose 4 different systems. In addition, the analysis records the testing artifacts of those repositories, by categorizing and aggregating the tests of the microservices gathered. The analysis of Paper E then provides an overview of the testing architecture in MSAs, as well as the comparison of the testing architecture with conventional best practices for testing.

1.3.6 Mining Job Openings

Finally, paper G aims to demonstrate an aggregated view of the roles, responsibilities, technical skills, and soft skills that the industry requires from microservices practitioners. To do so, job-ads are gathered, sampled, and analyzed. First, a pilot step takes place where 25 job-ads are gathered with purposive sampling and then analyzed. Then, an automated script is used to gather the most recent job-ads advertised in GlassDoor. Specifically, the script gathers up to 30 job-ads a day from each of the selected countries. In paper G, the automatically mined data of job-postings follow the inclusion criteria to include job-ads from English-speaking countries across all continents.

The gathered data are sampled and analyzed using techniques from GT. Specifically, stratified sampling is conducted to include a representative set of job-ads in the analysis, from all the countries of origin. Specifically, a selection is made using random sampling, proportional to the number of job-ads gathered for each country. In addition, Paper G uses a mixed methods analysis. The method includes on the one hand a manual analysis step using GT techniques to derive responsibilities and skills. On the other hand, quantitative analysis indicates the set of skills (both soft and hard skills) that job-ads require.

1.3.7 Threats to Validity

The research methodologies of the thesis inherit some threats that should be taken into consideration.

Internal Validity First, a threat to internal validity comes from the pre-exposure to existing literature and practices in MSAs. Specifically, existing knowledge on MSAs at the time of designing and conducting the research may have biased the design of interviews and data gathering from StackOverflow. In addition, it cannot be claimed for the resulted taxonomies, processes, and lists of activities to be exhaustive, since they are limited from the data gathered. Furthermore, the contents shared by the interviewees, the StackOverflow community, and job-ads represent the perception of the data-source, and it is arguable to what extent perceptions align to the reality. To mitigate these threats, data gathering is designed on the one hand to be as non-intrusive and as broad as possible. On the other hand, meta-data that indicated the validity of gathered content was used (e.g., including StackOverflow discussions that have an overall positive reaction score from other practitioners).

External Validity In terms of external validity, it cannot be guaranteed that the data populations of this thesis represent the entire software industry. For example, interviewees are sampled using a voluntary procedure, StackOverflow data represent only the parts of the software industry that are active in Q&A communities, and the gathered job-ads represent the market of the period that they were gathered. To mitigate this threat, it was attempted to ensure diversity in the data population, in terms of organizations size, industry domain, and geographical regions. In addition, the usage of multiple data sources for triangulation and conducting different studies helps to mitigate this threat.

1.4 Contributions

This thesis has seven contributions. Figure 1.3 gives an overview of how the different outcomes compose each contribution of this thesis.

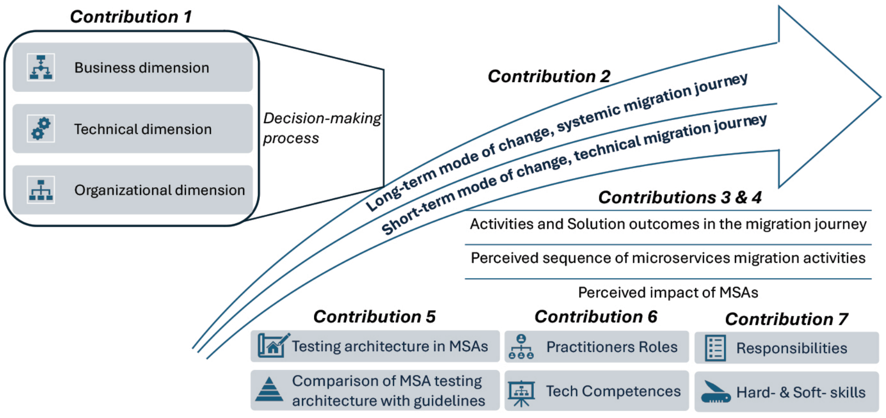
Contribution 1
Business dimension
Technical dimension
Organizational dimension
Decision-making process
Contribution 2
Long-term mode of change, systemic migration journey
Short-term mode of change, technical migration journey
Contributions 3 & 4
Activities and Solution outcomes in the migration journey
Perceived sequence of microservices migration activities
Perceived impact of MSAs
Contribution 5
Testing architecture in MSAs
Comparison of MSA testing architecture with guidelines
Contribution 6
Practitioners Roles
Tech Competences
Contribution 7
Responsibilities
Hard- & Soft- skills

Figure 1.3: Overview of contributions per research question

1.4.1 Contribution 1: Decision-Making Process

The first contribution of this thesis is the proposed decision-making process that covers comprehensively the multidimensionality of microservices migrations. This contribution is mainly covered by paper A, which reports a study of decision-making in microservice migrations by organizing aspects that describe past migrations. The research methodology of the study is GT-based interview analysis. Special emphasis is given to the human aspects of migration.

After analyzing the first 5 interviews, it became evident that similar tasks were perceived and executed in different ways across different cases and different engineers. In addition, interviewees mentioned that they had to think, deliberate, and make choices, at times, that later on were influential for their migration. Therefore, it became apparent that not only the migration process was central, but also the decision-making process in migrations. Therefore, it started becoming interesting to investigate in detail whether interviewees considered other options as well, how they made their choices, and why.

Consequently, in the interviews analysis, a mechanism was devised to uncover implicit decisions that software engineers make. There were three predominant ways of identifying decisions in our interview material. First, when the interviewees mentioned that they had to decide between different alternatives. The second way was when interviewees seemed unsure about a choice they had made and discussed the rationale behind it or chose two options simultaneously. The third way was when we identified different courses of action taken from different interviewees for the same task at hand. Decisions in all different dimensions were influential for the overall course of the migration. Therefore, there is evidence of decisions influencing decisions of other dimensions.

In paper A, the decision-making processes that happen on all levels of a microservices migration project are charted holistically including 22 decision points. This helps us understand the architectural design decisions in microservices migrations and how they tackle surfacing challenges (business, technical, and organizational). Also, it enables us to aggregate the migration journey and provide a framework for navigating this changing journey. A strong emphasis is given to the multidimensional nature of migrations towards microservices, considering the business and organizational side, as well as the technical side. In paper A's theory, we present 3 main dimensions: the business, technical, and organizational dimensions.

In the business dimension, the developed theory reports the need to first create engagement across the organization for migrating to an MSA. Specifically, engineers who know the need for migrating have to propagate this knowledge to other key stakeholders and engage them. It is not possible to just pull the plug and change the system at once. The identified decisions were first, on how to assess feasibility and explore potential opportunities. These decisions feed information into the development of a business case that drives the development of a new architecture.

The second dimension is about decisions on technical aspects. Specifically, in this dimension we chart different choices that engineers have to make when migrating. On the one hand, some of those choices are about grand design decisions of the migration like what splitting strategy to use or at what granularity should splitting stop. On the other hand, choices made in this dimension are very specific technical details like how to reuse and how to expose code.

Finally, a decision-making process is reported on the organizational dimension. In the third dimension, engineers were often involved in decisions that are regarding the organization and structures of the company. One theme in this dimension is decisions on the way the organization's operations change. Another theme is regarding rethinking the structure of the software development organization. Finally, some engineers had to work on deciding how knowledge is shared across teams.

1.4.2 Contribution 2: Migration Journey towards Microservices

The second contribution of this thesis is an iterative process for microservices migrations, presented in paper B and paper C. This contribution showcases that migration projects are continuous improvement initiatives instead of one-off projects. The findings of the studies deriving the migration journey organize different aspects that describe past migrations and present them in a continuous endeavor that takes place in iterations. Understanding the progress of migration projects in the aggregated process can help engineers gain awareness of the progress of different modes of change. Also, the charted migration journey showcases the different paces in different modes of change.

For example, paper B presents two main modes of change during a migration. The first mode is on changes in the software architecture and thus, it is about long-term changes and architectural design decisions - this mode is mapped to the systemic migration of paper C. The second mode is about specific system updates that are more operational, taking place in smaller sprints (software system-level migration) - this mode is mapped to the technical migration of paper C. In these modes of change that we identified, there are re-occurring phases. The phase of making design decisions is about the design activities that take place at the start of a migration sprint (architectural or system-level). Then, the phase of altering the system is about the implementation activities that actively modify the software application, on the different modes of change. Finally, the phase of implementing additional technical artifacts is about the development or modification of software or other artifacts that are needed along with microservices.

Building on the modes of change and the migration phases, the analysis of an additional dataset confirms the initial theory and modifies it accordingly. Specifically, the aggregated migration journeys of 16 organizations are complemented through the analysis of 215 posts from StackOverflow, that discuss microservices migrations. The analysis of StackOverflow discussions captures the content that is shared among developers, often including information on the ways that they work, think, and tackle different issues regarding their microservices migrations. StackOverflow is often the place that software engineers turn to when they face challenges in their work [53]. The StackExchange data explorer [1] is used to collect what developers discuss when transitioning to microservices. The two modes of change are updated into the systemic migration on the one hand and the technical migration on the other hand. In addition, the additional dataset directed a different distinction between the phases, resulting in the merging of two of them. A more general *Planning* phase of each migration iteration is followed by an *Execution* phase that is followed by a phase for *Setting up supporting artifacts.* The phase of setting up supporting artifacts is a stage in the migration where the development and operations are configured to support effectively the new paradigm that microservices bring.

1.4.3 Contribution 3: The Constituent Elements of Migrations' Journey

In the third contribution of this thesis, paper C further develops and extends the theory on how migration journeys take place and describes the parallel modes

[1] https://data.stackexchange.com/

of change in more detail. The two parallel migrations intend to demonstrate the way in which organizations execute overall systemic changes and specific technical changes. These modes of change explain in detail specific activities that take place during migration iterations. Moreover, the detailed analysis of StackOverflow discussions defines the constituent elements of the migration process, across different levels of abstraction. Specifically, 14 activities are identified in total that all together have 53 different solution outcomes.

The systemic migration is on a broad and slow-paced scope, taking place on the global software architecture transition that is required when an organization commits to an MSA migration. The long-term vision of the systemic migration concerns mostly structural, organizational, and business aspects. For example, an activity in the planning phase is about clarifying drivers for migrating, which requires business-oriented input. Another example is that in the execution phase, the activity of designing a service cut concerns a structural change and the activity of setting up continuous extraction is about organizational change to facilitate the decomposition of the system.

The scope of technical migration is narrow and fast-paced, focusing on the technical realization of a migration towards MSA. Specifically, the short-term scoped technical migration concerns technical design decisions that are critical for the migration, but are very specific and far from the broader picture of the architecture. For example, splitting up the data is an activity that could be deemed irrelevant in the grand scheme of things, but it is still crucial in migrating the software to a new architecture. Another example is the activity of setting up monitoring, logging, and authentication, which is not crucial for the value-adding changes of the system, but the absence of such solutions might be highly costly.

The analysis of the 16 migration cases results in a pragmatic view of migrations towards microservices. The migration journey is what the software development organization and the engineers go through to achieve a relatively mature state of their microservices architecture. The suggested process iterates until the architecture, system, and work of engineers reach a final, stable state. During this journey, software development organizations came across several activities, tasks, and solutions that are identified, categorized, and listed. Existing research provides patterns that direct organizations on how to migrate towards MSAs, but the results of paper C show how activities of migrations connect and how they materialize into solution outcomes. Such findings contribute to forming abstract patterns into actionable practical activities with concrete solution outcomes.

1.4.4 Contribution 4: Perceptions of the Impact and Sequence of Activities in Microservices Migrations

This thesis contributes also with an investigation on the perceived impact and sequence of activities in transitions towards MSAs through Paper D. A survey study is conducted that enriches the charted migration journey of organizations towards MSAs, by providing a better understanding of how microservices migrations are carried out. Specifically, the findings from the survey demonstrate a common chronological order of migration activities that potentially exists, based on the perceptions of practitioners. In addition, this contribution highlights specific areas where practitioners perceive the impact of the migration process to be significant.

The analysis of 54 survey respondents (microservices practitioners) firstly empirically reveals the sequence of 6 migration activities. Specifically, practitioners expressed the tendency to perform *database refactoring, DevOps infrastructure setting up,* and *back-end refactoring* earlier than *front-end refactoring, microservices communications setting up,* and *teams reorganization or splitting.* The concluded order indicates an apparent prioritization of the first 3 activities. Hence, practitioners perceive the database, DevOps infrastructure, and back-end as being more important or more feasible to start with during a microservices migration.

Moreover, the analysis of the survey respondents demonstrates the perceived impact of microservices migrations, across 7 impact areas, as well as business and operational improvements. The 7 impact areas were presented in the survey and practitioners ranked them in comparison to each other. The responses demonstrate first that practitioners perceive a positive impact of MSAs in respect to *aligning teams' structures with profitable value propositions*, as well as *testing process improvements.* Furthermore, a moderate impact is expressed on having *less dependencies between teams* and *improving the overall agile development process* of teams. Surprisingly, more practitioners expressed that MSAs impact in *structuring teams based on system parts* (service ownership based on functional concerns), rather than *structuring teams based on features* (service ownership based on end-to-end features). Respondents indicate that MSAs do not necessarily *alter the alignment between software and business.* However, respondents agree that several business and operational improvements have resulted from MSAs. For example, 67% of respondents agreed that MSAs enabled improved quality in the experience delivered by their software. In addition, more than 50% of the respondents indicated that MSAs improved scalability and maintainability of their software.

1.4.5 Contribution 5: Empirical Analysis of Testing in MSAs

The fifth contribution of the thesis is through an empirical investigation of testing in MSAs. Specifically, Paper E provides an overview of the testing architecture of microservices software. The testing architecture is devised from the analysis and categorization of artifacts in GitHub repositories of microservices software. In addition, an analysis is conducted to assess how the testing architecture aligns with testing guidelines and specifically the test pyramid. The focus is on microservices systems that adopt consumer-driven contract testing (CDC) since such systems are suitable due to adopting several principles of MSAs. CDC is a way of safeguarding the integration of several microservices and this contribution provides evidence and observations to the body of knowledge on how CDC is being used in practice.

The derived testing architecture from analyzing the mined microservices classifies tests as unit, integration, component, and system tests. The identified testing types include the testing artifacts that are used in the studied system, the relations between those artifacts, and the tools or frameworks that implement testing designs. One finding of the analysis is that mocking artifacts appear in all testing types. Furthermore, it is apparent that component and integration tests are often used interchangeably, and component tests cover integration, even though they do not always target the interaction between microservices.

Moreover, the analysis is extended with a comparison of the identified testing artifacts to testing guidelines. Specifically, the number of test cases of each test type is aggregated, and the proportion of tests is compared with the test pyramid. The size of the system plays a role in the alignment of the test artifacts with the test pyramid. Specifically, in smaller systems, the pyramid shape is not very distinctive, whereas in larger systems the pyramid appears more distinctly. Nevertheless, the distinction between integration and component tests is not clear in most microservices, indicating the need to introduce a different categorization of testing in MSAs.

1.4.6 Contribution 6: The Roles and Technical Competences in MSAs

This thesis contributes further with the identification of the roles and technical competences of practitioners working with MSAs in Paper F. Specifically, a large-scale StackOverflow analysis investigates the different existing types of profiles that work with MSAs. First, this contribution proposes a taxonomy that articulates the technical characteristics of software engineers working with microservices. Then, the profiles are stratified on the derived taxonomy, and their primary and complementary competences are identified, demonstrating an overview of what technical competences engineers working with microservices typically have. The findings from analyzing StackOverflow profiles and tags indicate the job competences of microservices engineers, potentially helping developers on what technologies are important in the context of MSAs.

The technical competences taxonomy of microservices practitioners groups in competences collections the 11 identified technical competences clusters. The 11 technical competences clusters are automatically identified (from the Leuvain clustering algorithm) and their contextual themes, as well as their grouping, are determined through manual analysis. The taxonomy entails the 3 overarching competences collections of Web Technologies, DevOps, and Data Technologies, organizing 8 competences clusters. In addition, there are 4 stand-alone competences clusters. The competences collection of Web Technologies has competences in API development and service integration, full-stack development, front-end development, and security and networking. In addition, DevOps has version control and quality assurance, as well as monitoring and CI/CD. The Data Technologies competences collection has data analytics and engineering, as well as data management. The remaining competences clusters are stand-alone, and they are about programming, data structures and algorithms, mobile development, and Microsoft-specific cloud services.

Moreover, in this contribution the association of how profiles of microservices practitioners with the 3 overarching competences collections is calculated. The analysis indicates that the majority of microservices practitioners have competence in Web technologies and DevOps, whereas almost half have competences with Data technologies. In addition, Web technologies are more often the primary competence of practitioners, complemented by DevOps and/or Data technologies. Specifically, most microservices practitioners with web technologies competences have some DevOps competences as well.

Furthermore, the roles of microservices practitioners are described and characterized. The roles are defined as Web-based software engineers, DevOps engineers, and Data engineers. The tags associated with practitioners indicate the primary and complementary competences. Microservices practitioners seem to either have a predominant specialization or a comprehensive technical skillset with multiple competences. Specifically, competences clusters such as API development and service integration, monitoring and CI/CD, or data analytics and engineering tend to call for exclusive specialty, whereas other competences clusters tend to more often be complemented by other competences with high association.

1.4.7 Contribution 7: The Responsibilities, Soft-skills, and Hard-skills of Microservices Practitioners

Finally, the last contribution of the thesis is a comprehensive overview of the responsibilities of microservices practitioners, as well as the skills (both soft skills and hard skills) that practitioners need for MSAs. Specifically, Paper G gathers and analyzes public job-ads. The job-ads analysis confirms and extends the roles of microservices practitioners. Moreover, the responsibilities of microservices practitioners are identified, focusing either on the product artifact, or the software development organization, which seems to be highly human-centric. The separation of responsibilities provides an initial understanding of the different aspects that describe microservices practitioners, allowing the investigation of those aspects separately. Additionally, the findings from analyzing job-ads provide details on the skills of microservices practitioners.

In this contribution a taxonomy of 21 responsibilities, organized into 5 families of responsibilities is identified. Human-centric families of responsibilities are found in Software process and team development, Professional services delivery, and Governance of software engineering. The human-centric responsibilities are concerned with the software development organization in relation to the software engineering team or customers. The fact that all but a few job-adds of microservices practitioners include human-centric responsibilities, indicates the increasing importance of human-centered responsibilities and skills in MSAs. Other families of responsibilities are software development support and infrastructure and software product delivery. These responsibilities are concerned with the software product in relation to the engineering team or customers.

The skills taxonomy of microservices practitioners entails 8 soft-skills, 11 hard-skills, and the relationships of hard- and soft- skills. From those skills, the 5 most demanded soft-skills are Articulation and transferability of knowledge, Stakeholder management, Problem-solving, Communication, presentation and negotiation, and Leadership. In addition, the 5 most demanded hard-skills are Quality assurance, Cloud Infrastructure, Data Management, APIs and event-driven architectures, and CI/CD engineering. Furthermore, a calculation estimates how many more soft-skills on average job-ads have when a particular hard-skill is included. This reveals which hard-skills tend to occur alongside more soft-skills on average, indicating that human aspects play a role in utilizing some hard-skills in practice.

1.4.8 Addressing the Research Objectives and Questions

The findings from the seven contributions presented compose the answers to the thesis' RQs.

Answering Research Question 1: The decision-making process of organizations during a microservices migration is the process of deliberation for business, technical, and organizational decisions. The identified decision-making process entails 22 decision points, along with the point in the migration that decisions are made. Additionally, the alternative options of the decisions are aggregated from the interviews.

Answering Research Question 2: The migration journey that companies go through when transitioning to microservices is the iterative process of continuously improving the architecture. The journey takes place in two modes of change, a short-term technical mode and a long-term systemic mode. In addition, the iterative process has the phases of planning, implementation, and setting up supporting artifacts, in which the change takes place.

Answering Research Question 3: The constituent elements of migration journeys are migration activities, as well as solution outcomes that result from these activities. Specifically, 14 migration activities and 53 solution outcomes are presented. In addition, a subset of the identified migration activities are ordered chronologically.

Answering Research Question 4: The testing architecture of applications that use MSAs covers unit, integration, component, and system-level testing, with mocking being used at all levels. In addition, the boundaries between integration and component testing seem to be unclear in practice and service-level tests become more prominent.

Answering Research Question 5: The impact of MSAs on the organizational structures of migrating enterprises is the business and operational improvements that microservices introduce (according to impact factors investigated). In addition, microservices migrations call for changes in organizational aspects such as organizational structures and processes, as derived from the systemic mode of change in the migration journey.

Answering Research Question 6: The roles of practitioners working with microservices are Web-based software engineer, DevOps engineer, and Data engineer, as derived from the topics that practitioners show proficiency in StackOverflow. The responsibilities of microservices practitioners are on the one hand human-centric, regarding the software engineering organization and governance. On the other hand, responsibilities are regarding the software artifacts produced for teams or customers.

Answering Research Question 7: The technical competences and hard-skills of microservices practitioners are centered around the roles described in RQ6. Some examples are API design and development, quality assurance, and infrastructure configuration. Important soft-skills of microservices practitioners are around the articulation and transferability of knowledge, stakeholders management, problem-solving, and communication, presentation, and negotiation.

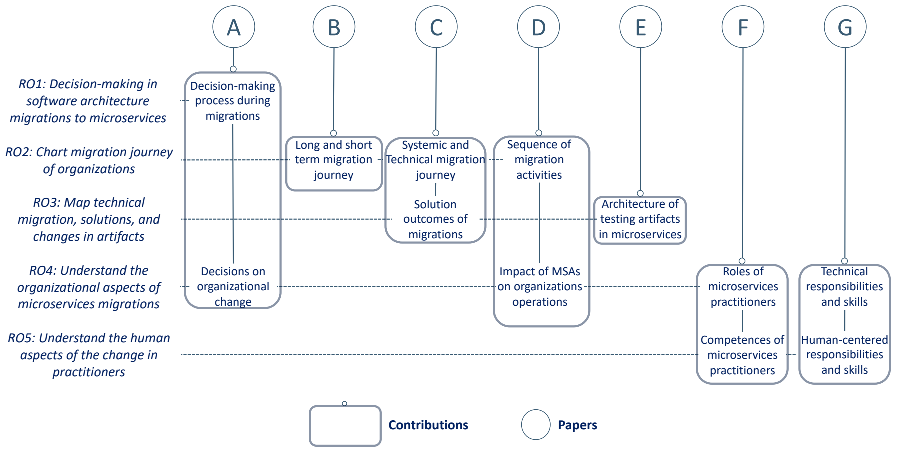
A
B
C
D
E
F
G
RO1: Decision-making in software architecture migrations to microservices
RO2: Chart migration journey of organizations
RO3: Map technical migration, solutions, and changes in artifacts
RO4: Understand the organizational aspects of microservices migrations
RO5: Understand the human aspects of the change in practitioners
Decision-making process during migrations
Decisions on organizational change
Long and short term migration journey
Systemic and Technical migration journey
Solution outcomes of migrations
Sequence of migration activities
Impact of MSAs on organizations operations
Architecture of testing artifacts in microservices
Roles of microservices practitioners
Competences of microservices practitioners
Technical responsibilities and skills
Human-centered responsibilities and skills
Contributions
Papers

Figure 1.4: Overview of contributions per research objective

As presented in Figure 1.4, the contributions of each paper help to address the research objectives of the thesis, as they are described in Section 1.2.1. This takes place by answering the research questions described in Section 1.2.2. For example, Obj-1 (understanding the decision-making process of microservices migrations) is addressed by answering the equivalent RQ1 through contribution 1 which provides a comprehensive decision-making process.

Moreover, Obj-2 (charting the migration journey) is addressed through RQ2 and RQ3, via contributions 2, 3, and 4. Specifically, contribution 2 indicates the long-term and short-term mode of change in a microservices migration. In addition, contribution 3 distinguishes and details the overall systemic change that takes place from the technical change that takes place during the journey to microservices. Finally, contribution 4 adds the aspect of sequence to migration activities.

The objective Obj-3 (aggregating technical migration solutions and changes in artifacts) is addressed through RQ3 and RQ4. Firstly, contribution 3 aggregates the solution outcomes that engineers result to during their migrations to microservices. Then, contribution 5 describes in detail how testing takes shape in MSAs. In addition, a comparison is conducted in contribution 5, between testing in microservices software applications and general testing guidelines.

Furthermore, Obj-4 (organizational aspects of microservices migrations) is addressed from contribution 1, 4, 6, and 7, answering RQ1, RQ5, and RQ6. Specifically, contribution 1 provides the organizational decision-making process taking place in migrations. In addition, contribution 4 indicates the impact of MSAs on an organization's operations and business. As an in-depth analysis of the roles that microservices practitioners have, the results of contribution 6 is input to organizational structures based on the profiles of engineers. Finally, contribution 7 enriches the input to organizational structures with details on the responsibilities that roles have, as well as extended descriptions of identified roles.

Obj-5 (understanding the human-aspects of the change in practitioners) is addressed with RQ6 and RQ7 through contributions 6 and 7. Specifically, contribution 7 provides details on the competences that microservices practitioners typically have. Furthermore, contribution 7 enriches the existing body of knowledge about practitioners' competences by distinctly identifying and discussing the human-centered responsibilities and soft-skills that microservices practitioners are required to have.

1.5 Discussion

This thesis gives a strong emphasis in microservices migrations as a socio-technical endeavor. The first four contributions do not only approach migrations as a technical endeavor, but also as an endeavor with a strong social, organizational, and business aspect to it. Contribution 5 is unraveling the underlying structures of testing artifacts. In addition, the last two contributions investigate further the organizational aspect of MSAs, with a deep focus on the human-aspects that MSAs predispose. Consequently, the thesis is covering the basic elements of socio-technical systems as defined in literature [20].

1.5.1 The Socio-Technical Perspective in MSAs

Stakeholders sensing and comprehending the concerns of other stakeholders, in the context of MSAs. As defined by Baxter and Sommerville (2011), the sensitization and awareness aspect of socio-technical systems engineering is regarding achieving awareness of the concerns that stakeholders have, among other stakeholders. The established process, the organizational structure and the technical infrastructure, along with the inherited human-driven complexity in their relations [20, 37] is taken into account in the appended studies. The contributions of papers A, D, F, and G form a starting point for creating awareness in the context of MSAs. Specifically, the different dimensions of the decision-making process identified are describing and bridging the business, technical, and organizational decisions that take place in microservices migrations. In addition, the perceived impact of microservices migration activities indicates how the adoption of MSAs is perceived by stakeholders. Finally, the competences and responsibilities of microservices practitioners indicate explicitly the concerns of practitioners that hold specific roles.

Integrating the change process with the software engineering process of adopting MSAs. Another aspect of socio-technical systems design is that of constructive engagement which is regarding the integration of the systems engineering process with the change process [20]. Paper B describes the change process with modes of change and their phases, as described in the context of a socio-technical system, in order to achieve a migration towards microservices. In addition, the second and third contributions integrate the technical migration process with the overal systemic migrtion process. The technical migration has to do with the specifics of the technology through detailed solutions, whereas the systemic migration addresses the overarching change taking place in a broader context. The connection of the change process with the software migration process is visible across this thesis in different forms. On the one hand, software that evolves in large scale and complexity forces its underlying technical structures to evolve as well and support it, as presented in papers A, B, C, and D. On the other hand, the roles, responsibilities and skills of practitioners are also altered with time, and their human-centered nature is important to be considered, as presented in papers F and G.

The evolution of software and software engineers in MSAs. This thesis present the evolution that MSAs introduce to the software architecture and the practitioners. In terms of software architecture, the contributions indicate how intentions for MSA migrations are propagated into activities of engineers. The derived process of decision-making approaches change from the perspective of intended deviations in the operational model of an organization, through explicit decision-making. However, the study in testing architecture reveals change through a perspective of unintended deviations from existing guidelines (i.e., testing pyramid). In terms of the practitioners' ecolution, the contributions give a comprehensive overview of what practitioners need in order to conduct their tasks. Specifically, the roles and technical competences are defined in detail. Hence, training plans can take into consideration both hard-skills and soft-skills that companies need from microservices practitioners. Organizational and human-centric challenges are important for the adoption of microservices [22, 29]. Investigating the skills of microservices practitioners introduces a structure on the different experts work and interactions. This thesis extends indications of existing literature regarding organizational structures in microservices.

1.5.2 Software Architecture Perspective

Software architectures entail different meanings for different stakeholders at different points in time. A substantial part of this thesis ultimately describes how organizations migrate towards a MSA. Both practice and academia provide varied directions on what a software architecture is exactly and thus, it is natural to end up with such a diverse set of aspects that describe the change and migration of software architectures. Seminal literature in the topic, indicated from early on how software architecture is different for different people, with the different potential views of a system's architecture [57]. The evident multidimensionality of this work strengthens this view on the topic of migrations to MSAs as well.

To complicate the scoping of the topic even further, software architectures have different utilities at different points in time, as software evolves. This is present in paper C, and there is related literature that supports these findings as well [58]. Paper C, can indicate to practitioners what software architecture is in different phases across time. At the start of development an architecture can be described as an imaginary design, helping to form a plan. During the development it can be viewed as a framework to share a common picture of how the system is and execute the migration on the system. After development is a navigation map to direct engineers on where each part of the supporting structure is. In terms of theory, software architecture was initially perceived as the formal structures that act as foundations of software systems. Hence, software architectures started to be perceived as a representation of a software system at a current state. However, the static nature of such an explanation came quickly in conflict with the dynamic and iterative nature of engineering software [59].

MSA migrations entail decisions and communication from engineers all the way up to executives. The contributions of this thesis provide links between narrow technical details with broad organizational structures. Software architecture is used to communicate the constituent elements of a software application in different levels of abstraction. Architectural representations of a software application compose the artifacts that describe the shared understanding between stakeholders. However, as development evolves, there can be a difference between the representation and the actual implementation. This does not mean that there is no value in architectural representations, since they can be used to navigate systems, but also have an overview of the big picture [57]. Papers B and C build on these ideas and draw empirical results along the lines of propagating the change on different levels of detail.

In addition, paper C goes even further on these concepts and demonstrates the solution outcomes and different activities of the development of the architecture when migrating. Specifically, the contributions demonstrate showcases that there are things that matter at all levels of detail and levels of abstractions. Narrowly scoped technologies and tools influence large, systemic design choices and vice versa - overarching design choices influence specific technical solutions. Analyzing and altering the software architecture takes place on all levels, from strategic to coding, as indicated also in paper A with the business aspect of migrations.

1.5.3 Breaking Down the Complexity of MSA Migrations

Researchers should put more focus on the non-technical aspects of migrations. MSA migrations contain many clusters of sub-topics, ranging from changing source code, to modifying the business service delivery mode. Literature and practice discuss software architecture migrations with the perspectives of source code decompositions, testing, integration and deployment approaches [6, 12, 13, 16]. This diversity indicates the complexity that exists when migrating towards a MSA, as do individual studies that investigate instances of such migrations [14]. However, topics that do not have a technical focus are not investigated extensively and yet, they are important. The results of this thesis specify further the individual elements of MSA migrations and break down the known complexity of such endeavours. Especially papers A, B and C indicate the different aspects of MSA migrations and give details on them. Researchers and practitioners can benefit from this research with scoping migrations and linking them with the overall picture that the results present. Moreover, the contributions from papers F and G are a starting point in bridging the gap between the technical aspects and the human-aspects of the practitioners' work in MSAs.

Migration drivers, business and organizational needs, and human factors need to be considered when planning migration processes. The business drivers cannot be ignored when migrating, since they fuel the actual change at scale and across the organization. Migrations of software systems and technologies can happen to align the software architecture with the overall service delivery strategy of organizations and thus, help achieve business objectives [19]. Clarifying migration drivers complements current research that describes the benefits of migrating towards microservices [33, 60], or away from microservices [10]. Benefits and drawbacks are technical, performance-oriented or economic, with details on how to align different stakeholders.

Additionally, software architectures with modern, cutting edge technologies can help establish socio-technical systems that contribute to the required organizational agility for being competitive in modern ever-changing economies [2]. The derived dimensions and modes of change in this thesis, provide a comprehensive view, adding to existing research ways to engineer a microservices migration within the organization. Also, the findigs can help practitioners with making choices that are not only driven by technical limitations, but also by what the business/customers need, what the organization needs and what the technology can facilitate. Hence, this research consists of a critical view on when it is possible and useful to migrate and until which level of migration completion.

1.5.4 Decision-Making in MSA Migrations

Future work should investigate individual and group decision-making processes. It is well known that changing the software architecture is substantial and can involve many decisions [58]. A decision-making process is therefore important as also indicated by other studies [18] and paper A investigates decision-making from many different organizations. Decision-making of software engineers takes place in an individual level [38], in groups/teams or small software organizations [40] and in large organizations that develop software [61]. Paper A takes a perspective of organizational decision-making, showcasing a multidimensional approach that involved the perspective of business change, the perspective of technical change and the perspective or structural change on the organization. The results of paper A complements the existing state-of-the-art, since according to Hassan et al. [16], existing research investigates migrations as a technical endeavor that needs a technical solution, and paper A gives a decision-making perspective. However, so far there is few studies investigating individual or group decision-making in MSA migrations [18].

Decision-making processes are often implicit and can be extracted from engineers descriptions of their work. Moreover, the derived decision-making process in paper A showcases how future work can derive engineers' decisions from qualitative data. On the one hand, all interviewees when asked to describe their migration journeys, started by describing their course of action (i.e., "we first did 'a' and then 'b' and afterwards modified our approach..." and so on). This provided evidence of a sequence that each case followed in order to achieve the migration. On the other hand, more experienced engineers described some prerequisites that needed to be in place on a company-wide level, to facilitate the migration. In the same way, they also described other things that change when a migration matures. Deriving the decision-making processes of migration initiatives in software development organizations can help to better understand such transitions and help achieve their realization by design rather by coincidence.

1.5.5 Implications on Software Engineering Teams that Migrate to Microservices.

Practitioners should take into account the diverse skillset required in MSA migrations, when preparing to commit on a migration project. The different modes of change presented as well as the different dimensions of decision-making indicate the diverse skillset that is required by teams, especially since microservices predispose designing the business and the software at the same time [4]. Paper A showcases that the business aspect is critical in order to fund a migration and usually the justification for migrating is not technical, but business-driven. Additionally, both papers A and C touch upon the overall changes that take place in the operational model of organizations that migrate. Changes that require a strong understanding of how the organization is structured and how it can potentially change. Therefore, it can be argued that business-savvy software developers and programming-savvy business analysts and system designers are needed in teams to accommodate all perspectives and ways of thinking.

In MSAs, complexity is shifted from the software implementation to the configuration and integration of services. On the one hand, this is observed in paper B and C, where many additional technical tasks are needed just for supporting microservices. Hence, a big proportion of migration activities have to do with setting up the development process of microservices, their deployment, testing and integration. On the other hand, since integration of microservices plays such an important role for the development of the system, the communication between microservices can contain sometimes more business logic than the source code. An interesting result from this thesis is on the perception that there is strong decoupling in microservices. The reality in practice is that often a chain of microservices exists that brings coupling on the configuration level rather than on the source code level.

Migrations to MSAs often take place in parallel with maintaining, extending and growing the system under migration. Furthermore, this work indicated how many of the investigated software development teams that migrate do not consider the change of the system as their main value-adding project. Rather, they view the migration project as a necessary sideline activity and they focus on developing new features and value adding artifacts at the same time. Hence, migrations take place in parallel with other activities and thus, there is sometimes a pause and revisiting to the project, explaining partly their often iterative nature. Even in cases where dedicated personnel or teams take responsibility of the migration, there is a sense of parallelization with further development of the system. This can indicate to practitioners how broad the change can be in the organization. Specifically, the nature of the change touches many different parts of the organization that needs a broad synergy to make progress.

There is further need for future research to specify the scope of the investigated change. We distinguish decomposition of services into developing a shell API (similar to existing patterns [13, 15]), designing service cuts (relating to services designs [34,62]) and continuously re-extracting services (relating to designing MSAs [18]). The developed process frames them into the appropriate scope to investigate such topics in different stages of the migration. Current research touches upon designing MSAs, and this work combines parts of this knowledge in the systemic journey, putting the different activities in an accumulated perspective. In this accumulation, researchers and practitioners can obtain perspective about the proportion of these (seemingly important) activities in the overall migration and investigate the design decisions that take place. Related literature does not specify the scope of the investigated change in relation to the overall change that takes place and the theories developed in this thesis enables this.

1.5.6 Future work

This work paves the way towards understanding microservices migrations and their underlying decisions. The insights generated for migrating MSAs can be transferred also to other types of software architecture migrations. For example, to software-based systems that start as ad-hoc solutions and grow in scale. Such systems eventually need a rigid software architecture to support them and thus they are transitioned to new structures. Consequently, future work includes the investigation of other types of software architecture change due to scaling requirements. Specifically, we can study further the process of transitioning other cutting edge technologies into scalable architectures. For example, investigating how the wave of AI applications is integrated and deployed to existing systems that operate on a large scale. In addition, next steps include to investigate further the evolution of specific elements of the software architecture. For example, in future work it is intended to investigate how data handling, logging and monitoring changes in MSAs.

Moreover, in the future there is a need to investigate in more detail decision-making in designs for migrations and/or microservices architecture. While the purpose of this work is to understand empirically the "as is" process, the results could be seen as a first step towards providing decision support for software architecture migrations, as done in other areas for software engineering (e.g., requirements engineering, COTS selection). For example, a migration towards microservices can have many benefits to different stakeholders and future work can aim to comprehensively present the value delivered to the organization through all stakeholders.

Finally, more investigation is needed on the decision-making processes and the approaches to resonate about alternative choices. Hence, the focus of future work needs to not only provide knowledge about the outcome of decisions, but also on identifying the reasoning behind those decisions. Future research can also target the evaluation of such detailed decision-making processes. This indicate towards further work that is needed on individual decision-making and judgement.

1.6 Conclusion

As software systems grow large, both in size and complexity, it becomes difficult to maintain and update them. Implementing a MSA is becoming popular in different sectors for modernizing large or growing software systems. Existing literature reports well the value to move towards microservices [6], but the process to achieving a migration towards a MSA is not well known. Migrating towards microservices and changing the software architecture is a highly complex task that takes time [14, 15]. Specifically, it is not always clear how aspects of migrations connect to each other and how migration activities take place in relation to one another [16]. However, such migration projects entail an inherent complexity due to the different dimensions that the change takes place in, as well as the distributed nature of microservices. In addition, migrations to MSAs are often investigated with a focus on the technical change. The human-aspects that are in interplay with MSAs require further investigation, as in other sub-domains in software engineering [47–49]. The seven contributions of this thesis cover the basic aspects of a socio-technical perspective to change, using concepts from established literature on socio-technical systems engineering [20].

This thesis sees migrations in the light of multiple dimensions (business, technical, organizational), on multiple levels of abstraction (architecture and system) and in multiple modes of change (technical and systemic migrations). The results can help in understanding microservices migrations and carrying this understanding over to future migration attempts. Migrations of software systems and technologies (e.g., towards microservices) happen on a multitude of dimensions, due to the inherent complexity and the socio-technical nature of organizations. Microservices migrations have a technical side that is extensively investigated. However, there is also an organizational side that is very important, especially since change across multiple parts of the organization is involved. The organizational side can involve structural aspects, responsibilities or engineers as well as operational/process aspects. Importantly, migrations also have a business and domain-specific side that needs to be considered. Since many critical decisions are taken in the business side, considering human factors is also of great importance.

This thesis also has a strong focus on the human aspect of a migration, through the engineers' concerns, skills and their tasks, being a part of the migration. Specifically, we investigate how software engineers and companies go through a migration towards microservice-based-architecture. We obtain an understanding on the different dynamics involved in their transformation. Finally, both the journey and the decisions identified can help software development organizations and engineering teams to anticipate what is up-coming in their migrations. To achieve this goal, the empirical research conducted attempts to derive inductively from engineers' experiences the details of software architecture migrations towards microservices. Additionally, techniques from GT are applied to gather and analyze textual information that engineers share in Q&A websites and specifically, in StackOverflow. Furthermore, a large scale analysis is conducted on profiles of engineers and their associated tags in StackOverflow. Finally, testing artifacrts of microservices repositories are analyzed, as well as job-ads that target microservices practitioners.

The contribution of this thesis is threefold. Firstly, the thesis charts how decision-making takes place in migrations towards microservices, via a comprehensive decision-making process. Secondly, the overall journey of migrating a software architecture towards microservices is derived, including two modes of change that have several phases, activities and solution outcomes, as well as how testing changes in MSAs. Thirdly, this thesis investigates the organizational aspect of software development and sheds light on the roles, responsibilities, and skills of microservices practitioners.