Thesis for The Degree of Licentiate of Engineering

Migrations to Microservices-Based Architectures - a Tale of Technical and Organizational Change

Hamdy Michael Ayas

Division of Interaction Design and Software Engineering
Department of Computer Science & Engineering
Chalmers University of Technology | University of Gothenburg
Gothenburg, Sweden, 2022

Migrations to Microservices-Based Architectures - a Tale of Technical and Organizational Change

Hamdy Michael Ayas

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

This thesis has been prepared using LaTeX.
Printed by Chalmers Digitaltryck,
Gothenburg, Sweden 2022.

To my grandparents Vasiliki and Kyriakos, that their humble beginnings of working in olive trees, coper mines, (bee) herds and barley fields sparkled a limitless inspiration and support to my education

Abstract

Background: As software systems evolve and scale faster than the foundations on which they are structured on, software architecture migrations to modern, cutting edge paradigms of development are becoming common. An example of such a paradigm is Microservices-based Architectures (MSAs). With MSAs, organizations can manage the complexity of their software and deploy individual pieces autonomously and independently. However, migrating towards microservices entails a lot of complexity. The evolution of the structures that a migration predisposes is multifaceted, with a socio-technical nature.

Objective: Therefore, this thesis aims to first of all understand the process in which decisions are made by engineers to migrate their software architecture towards microservices. In addition, this thesis targets to aggregate the migration journey of organizations that change their software architecture to microservices. Finally, it is demonstrated how an organization's operations implement different processes for software architecture migrations and development methodologies.

Method: The methodologies used in this thesis are mainly qualitative methods. Grounded Theory and Grounded Theory-based analysis is used on interview data as well as textual data that engineers share in Q&A websites (i.e., StackOverflow). Moreover, a case study is also included to understand how engineers adopt certain agile practices and guidelines via observations, a survey and interviews.

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). In addition, 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. Furthermore, as the outcomes from analyzing microservices migrations resulted to processes and taxonomies, the thesis demonstrates a reflective view of developers percpectives in adopting processes and guidelines.

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. Also, this work paves the way for studying further migrations to scalable cloud-based architectures and viewing them as comprehensively as possible.

Keywords

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

Acknowledgment

First and foremost, I would like to thank my supervisor Regina Hebig and co-supervisor Philipp Leitner. Your input and insights have been invaluable, in shaping the research of this thesis. Even more importantly, your coaching helped to develop my research acumen and I will be forever grateful for that. Looking forward to continuing our research journey.

Next, I would like to thank my examiner Prof. Miroslaw Staron, for the constructive feedback. I would also like to express my appreciation to the discussion leader of this licentiate, Prof. Davide Taibi for accepting the invitation to discuss about my research.

It is a priviledge to have university ordinances that ensure a supportive environment. Thus, I would like to express my gratitude to Agnetta, Wolfgang, Nir, Clara and everyone contributing to the PhD school. Also, thank you Richard, Robert, Palle, Eric and Philipp for the trust and leadership.

Special appreciation goes to my early line manager, Ivica Crnkovic for the kindness, inspiration and interesting discussions. You will be remembered.

Next, I would like to thank Francisco and Mohammad for the joyfull collaborations. Special thanks go to current and past office mates and friends, Joel, Linda, Razan, Georgios, Peter and Mazen, for the friedly and colorful working atmosphere. I also thank Richard B. S., Ricardo, Habib, Cristy, Jan-Philipp, Sjoerd, Yuchong and all collegues in the IDSE division for the nice time.

I am gratefull to Manos, Giannis, Vasiliki, Christina, Dimitris P., Penelope, Maria, Iosif, Eva and Dimitris T. for making Gothenburg feel like a home. You are all special and the best companions one could wish for. Additionally, I would like to thank Thomas and Christos for the inspiring pragmatism and our unique interactions. Santiago and Eirini, thank you for all the shared color, flavours, music and experiences. It is only the beginning of our journeys. A special place in my heart will always be occupied from my childhood friends, Kyriakos Pel., Kyriakos Pap., Dimitris, Paris, George, Andreas and the rest of the gang back in Cyprus. Thank you for all the fun summers and winters.

Most importantly I thank my parents - Mohammad and Eleni, my beloved siblings - Sandy and Malek as well as my beloved niece and nephew - Maria and Ali. You are my roots, my wings and my light.

Last but not least, I would like to thank my lovely partner 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. I am forever gratefull to life for your incredible and endless love, support, patience and positive energy.

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"
In submission.

[D] 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.

Other publications

The following publications were published during my Licentiate 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] 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

Research Contribution

I (Hamdy Michael Ayas) was the main driver and contributor of Papers A, B and C. In addition, I had major contributions in Paper D. A summary of the contributions is presented in Table 1, based on the Contributor Roles Taxonomy (CreditT) [1], as presented by Brand et al., 2015 [1].

For Papers A, B and C, 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, Validation, Visualization and Writing of the original draft.* I also facilitated the ways that co-authors (main and co-supervisor) contributed in the *Formal Analysis* (e.g., via data analysis guides), to ensure enough rigour 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.

For Paper D, I contributed with the *Formal Analysis, Investigation, Validation, Visualization, Writing of the original draft and Writing through review and editing.* The work of Paper D is based on a master thesis that I collaborated with, in which a re-analysis of the existing data gathered and a complete re-writing of the publication took place.

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

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

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

Contents

Chapter 1

Introduction

Software is dominating at a large scale the ways in which the world operates. Milions of lines of code are present in the heart of our work, leisure, transport, economies, and democracies. As Marc Andreessen famously wrote on 2011, "In short, software is eating the world"[1]. Software that evolves in large scale and increasing complexity, forcing its underlying foundations to evolve as well and potentially drift from the intended structure [2]. Additionally, software systems need to have the capacity of changing fast with the development of new applications, updating of existing applications and combining/aggregating applications or digital services [3]. Consequently, as software systems evolve and scale faster than their foundational structures, software architecture migrations to modern, cutting edge paradigms of development are becoming more and more common [4, 5].

An example of a modern, cutting edge paradigm of structuring systems and their development is Microservices-based Architectures (MSAs). Hence, organizations in many industries are increasingly adopting microservices technologies to structure their software [6]. By adopting such a service oriented architecture like microservices, organizations can manage the complexity of their software and their systems can deploy individual pieces autonomously and independently [7,8]. Therefore, organizations adopt a software architecture that complements the Agile methodologies of software development and facilitates organizational agility [9].

1.1 Background

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 [10]. 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 [11]. Hence, microservices have fine-grained interfaces (e.g., API endpoints) of independently deployable services [10, 12].

[1] https://www.wsj.com/articles/SB10001424053111903480904576512250915629460

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 [13]. Furthermore, business-driven development practices are critical to accompany agile practices that are often adopted by software development teams [9,11]. In addition, microservices follow cloud-native application design principles [10]. 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). Another characteristic of microservices is their persistence strategies, especially in managing their own state [11] and having (in principle) their own database [14].

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 [12]. The differences of microservices with SOA is not on their architectural style but rather on the implementation of the architecture. Specifically, microservices embrace, leverage and add on principles and patterns from SOA (loose coupling, service contracts etc.) [9].

In microservices, Domain Driven Decomposition/Design (DDD) is key [11]. 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) [13]. This way a MSA can facilitate organizations to achieve improved scalability, maintainability and reduced time to market [11, 15]. Finally, MSAs are used in new software systems, but very often, an existing system needs to be migrated to a MSA [10] and there is plenty that we do not know about migrations to microservices specifically. Hence, there is a need to study the process of migrating towards such a software architecture [16].

1.2 Problem Statement

Migrating towards microservices entails a lot of complexity [17]. Specifically, the evolution of the structures that a migration predisposes is of a socio-technical nature, comprising a technical, an organizational and a social aspect [18]. Furthermore, these socio-technical concerns in 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.

The value to move towards microservices is overall well reported [10], but how to achieve a migration towards a MSA is not so straight forward. That is because changing the software architecture to microservices is a highly complex task that takes time [19]. 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 [20]. Also, architectural migrations are heavy in decision-

making [21], either in an individual-level, team-level or organizational-level (i.e., company wide) [3].

Furthermore, migration processes are often un-structured and take place in ad-hoc manners (they are not systematic or methodical, but rather take place on a trial and error basis). They are also not well known since they are not always systematically recorded [10]. There is knowledge from practice and academia on how to technically enact MSA migrations, but it is rare that many migrations are aggregated to show engineers a more generic view of change and thus, engineers often learn along the way.

Also, existing solutions (e.g., through program decomposition) are often not addressing the decision-making of engineers in migrations and can lead to ad-hoc, disconnected processes from the rest of the organization/system. For example, applying automated software decomposition tools on source code [22], does not mean that the system is migrated to a MSA. There are more aspects than just the source code that change during a migration, such as integration and deployment methods, testing and many more [23, 24].

Migrations entail decisions for bigger changes than just a systems' upgrades. They are transformative on organizations as a whole [25], and we have a lesser understanding on the non-technical aspects of migrations than the technical aspects. Hence, there is a lack of approaches providing details on the operational choices that software development teams and organizations make in migrations [23].

Moreover, organizations do not only go through intended change in their operations (e.g., MSA migrations), but also through unintended change [26]. Unavoidably, there are many practices that direct organizations on how to evolve in new operational paradigms, with change that is explicitly prescribed [27], but also change that is implicitly imposed during time [28]. Consequently, there is a deviation between propsed guidelines and practical activities that software engineering teams actually do. This deviation needs to be investigated, both in times of change (i.e., a migration), but also in times of stability (i.e., methodology during software development).

For example, adoption of agile practices is not as straight forward as it seems to be, even with frameworks like Scrum [29]. There is evidence that 37,5% developers report to intentionally deviate from defined processes [30]. Also, teams customize best practices to their particular needs [31]. It is not always clear what are the reasons for deviating from agile practices and the implications that deviating could cause.

1.3 Research Scope

To address the challenges described in the problem statement, this thesis has a set of research objectives and research questions. An overview is presented in Figure 1.1 of the research questions and how they relate to each other. Figure 1.1 demonstrated also the input of each research question and to which objective it contributes.

The following research objectives are covered in this thesis.

Research Objective 1: Understand the process in which decisions are made by engineers to migrate their Software Architecture towards microservices.

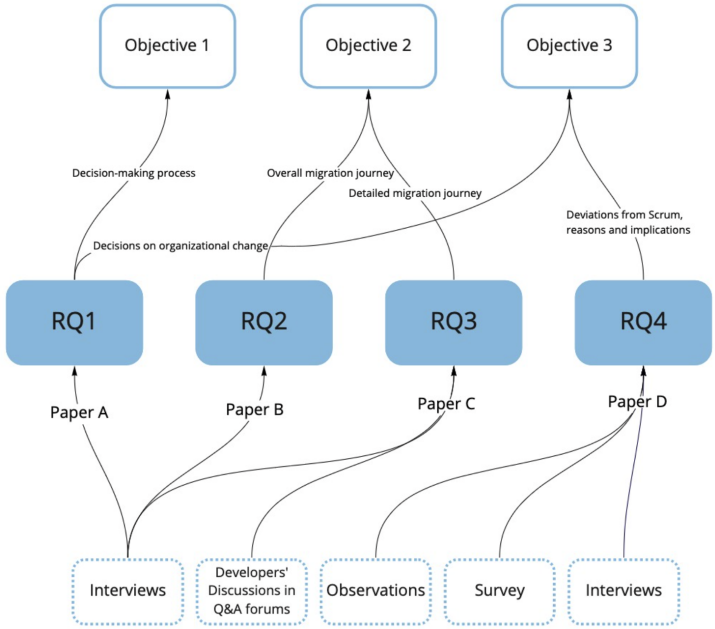

Figure 1.1: Overview of Research questions, the products they use and their outcomes

Research Objective 2: Aggregate the migration journey of organizations that change their Software Architecture to microservices.

Research Objective 3: Demonstrate how an organization's operations implement different processes for Software Architecture migrations and software development methodologies.

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? In this research question, it is intended to empirically derive the decisions that organizations make during a migration towards microservices and when these decisions are made. In addition, with this research direction it is intended to showcase the typical options that organizations can choose in such decisions.

RQ2: What is the migration journey that companies go through when transitioning towards microservices? This research question aims to address the gap on empirical understanding of migrations from the engineers' point of view. It is intended to investigate on what different levels can the migration journey take place and how these different levels are structured.

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 attempts to pinpoint the different activities that take place in different levels of abstraction accross the organization. Specifically, activities and common solutions are identified on the technical and systemic level of a migration.

RQ4: What are the deviations that exist in adopting processes and guide-

lines for Agile practices (why teams deviate and what are the consequences)? Finally, this thesis aims at understanding how engineers adopt agile practices and specifically Scrum. In addition, the thesis aims to provide an understanding on what makes engineering teams to deviate from proposed guidelines and practices, as well as on the consequences that such deviations can have. Hence, this research question aims to find typical deviations from the Scrum framework and pinpoint their reasons and implications.

To answer the RQs of this thesis, four papers are appended - paper A, paper B, paper C and paper D. All four papers use empirical inductive approaches to address the selected RQs. The combination of the four papers cover the basic aspects of a socio-technical approach to change, using concepts from established literature on socio-technical systems engineering [18], as shown in Figure 1.2. Paper A describes the sensitisation and awareness aspect of socio-technical systems engineering. Paper B describes the change process, as described in the context of a socio-technical system, in order to achieve a migration towards microservices. Paper C covers the systems engineering process, that has to do with the specifics of the technology through detailed solutions. Finally, paper D touches upon the constructive engagement of engineers with guidelines and processes.

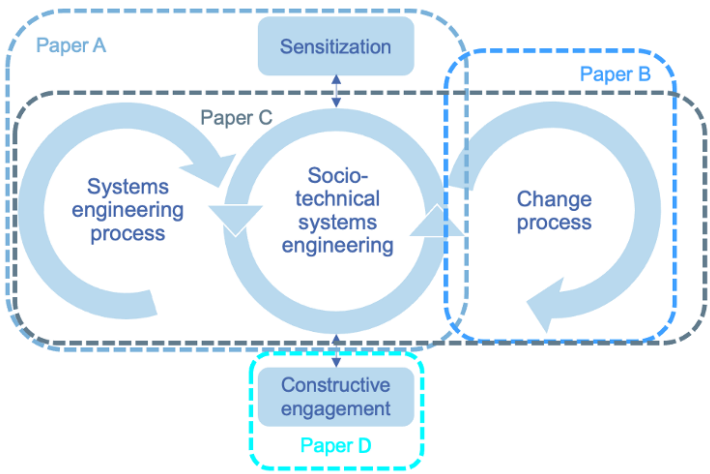

Figure 1.2: Overview of socio-technical systems engineering aspects coverage

1.4 Related Work

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 decision-making in software engineering.

1.4.1 Benefits of microservices

Often, software systems grow so large that it becomes challenging to maintain them and make new releases of features on them. Thus, 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 [32]. This difficulty of modernizing digital services is a challenge of organizational agility [9] and can also lead to obsolescence in products that are offered by organizations [33]. Hence, software systems need to be structured more meticulously and MSAs is a way to do so [15].

Microservices have many advantages. First, microservices are about focusing on one thing and doing it well [15]. Also, microservices enable faster releases of functionality and independent scaling and maintenance [10, 32]. The polyglot nature of MSAs enables technology diversity and faster adoption of different technologies, depending on the requirements at hand [9]. Additionally, a MSA enables separation of parts with high security requirements design and allows multiple points of failure to achieve resilience [34]. Some of the effects of microservices are response to business change, enable different workloads for cost improvements, higher value delivery, organisational agility, and decentralised governance [24].

1.4.2 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 [35]. 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 [16]. Such a re-structuring includes the evolution of monoliths to Service-Oriented-Architectures (SOA) and even further to microservices [9]. Monolithic software systems are tightly coupled pieces of software that have challenges in maintaining them effectively and updating them fast. SOA are a step towards organizing software with separation of concerns, functional and non-functional requirements and providing information as services [36]. 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 own individual domains and resources [9]. These individual pieces communicate with each other to compose a complete system [11]. Existing research investigates how to use business logic, domain and potential existing solutions [20, 24], but there is further room for relating —requirements engineering— specifically about microservices migrations.

—restructure the paragraph and combine with last— 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 [37]. 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 [34]. Also, there is a need in empirically investigating the details of migrations comprehensively from different points of view [35]. Migration projects are not simple, since migrating a system towards microservices (e.g., from a monolithic architecture) is a long endeavour with many things to consider and an inherent complexity [24]. Existing formal models can give guidance on how to track and split technical artifacts of the system [21,38]. However, there are not many empirical investigations on the process of designing microservices-based architectures.

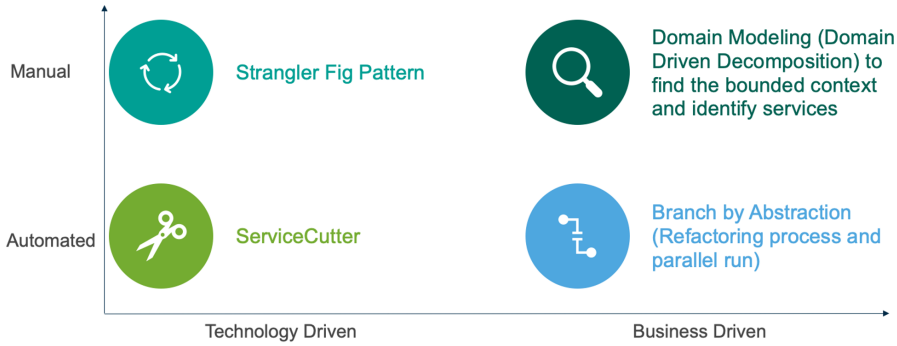

Figure 1.3: Categorization of microservices decomposition approaches

Research and best practices stemming from industry provide some approaches on migrations, covering many aspects [14, 24]. There are different ways for splitting the software to transform a system to microservices technically. Figure 1.3 showcases the landscape on which such approaches exist. Specifically, there are on the one hand manual approaches that deal with isolating through manual code analysis specific parts of the software to deconstruct services [24]. Such approaches have high awareness of context and include extensive human input. 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 [39]. Another way is meta-data aided, which is analyzing more abstract input data like UML, Use Cases, interfaces, commits etc. [40]. Also, microservices can be split using workload-data aided approaches [41]. Such approaches analyze measurements of operational data on 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.

1.4.3 Decision-making in software engineering

Currently, decisions for architectural designs are highly intuitive and based on previous experiences of engineers [42]. Software engineers have certain biases and limitations during cognitively demanding tasks [43]. Also, decision-making is especially challenging when it takens place in groups, since it is not common to have structured decision-making in groups [44]. Moreover, without having these humane particularities in the core of software engineering, it is more challenging to enable the full utilization of the engineers' creative and mental capacity [45]. These limitations along with their implications are extensively studied in many other fields but moderately studied in software engineering [46]. Also, existing decision-making processes do not comprehensively cover entire processes of software engineers and solutions are rather individual of a technology or a very specific problem.

1.5 Research Methodologies

In the research of this thesis mainly qualitative methodologies have been used. The methodologies used are mainly grounded theory (GT) or using techniques from GT as described in literature of using GT in software engineering [47]. Specifically, in paper A a GT study with interviews was conducted. During the interviews, the aim was to also capture descriptions and contextual information about the overall journey of the interviewees' microservices migration journey. Hence, it was possible to conduct another thematic analysis in paper B and derive empirically the migration journey towards microservices. Additionally, in paper C a methodology used was applying techniques from GT to analyze textual information that engineers share in Q&A websites and specifically, in StackOverflow. Moreover, in paper D we analyzed a case study to understand with rigour how engineers adopt certain agile practices and guidelines.

1.5.1 Interviews

The step of interview conducting and analysis is predominantly based on GT [47]. Specifically, the constructivist variance of GT is used. The starting point was an initial research question that evolved throughout the development of paper A, as suggested in literature for conducting studies based on constructivist GT [48]. The initial research question was further specified and broken down into what could be addressed based on the data analysis of paper A. A semistructured 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 qulitative analysis took place on the same interview-based dataset. The additional analysis step lead to the results of paper B and to the input/starting point of paper C.

1.5.2 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 [47, 48]. StackOverflow is often the place that software engineers turn towards when they face challenges in their work [49]. Therefore, discussions arise in this Q&A forum that contain a lot of details regarding engineers' concerns. Developers use posts from such websites to gather information, get ideas of solutions and discuss their design decisions to validate them [50]. More importantly, software engineers share issues and challenges they face [51], along with potential solutions of 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 [52].

This thesis, derives from (sometimes lengthy) discussions of software engineers in StackOverflow detailed solutions in migration activities towards microservices. The data collection includes the querying of StackExchange to gather questions that engineers posted as well as answers to those questions. 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.

1.5.3 Case study

This thesis also includes a case study, in order to understand how software engineering teams adopt agile practices, why they deviate and what are the implications of deviations from guidelines. The data collection of this case study used three methods. Specifically, observations were used to gather what exactly the software engineering teams were doing. Initially, the gathered observations were compared with the official guidelines of scrum and deviations started to be identified. Furthermore, the identified deviations were validated through a survey. Finally, a set of interviews were conducted in order to qualitatively derive indications on what caused the deviations and what was the implications of these deviations.

1.6 Summary of Contributions

The contributions of this thesis are threefold: 1) understanding the organizational decision-making of migrations towards MSA, 2) empirically infering how change takes place during microservices migrations and 3) describe how engineers adopt or deviate from guidelines of their used methodology. Figure 1.4 gives an overview on how the different outcomes contribute in asking each of the research questions of this thesis.

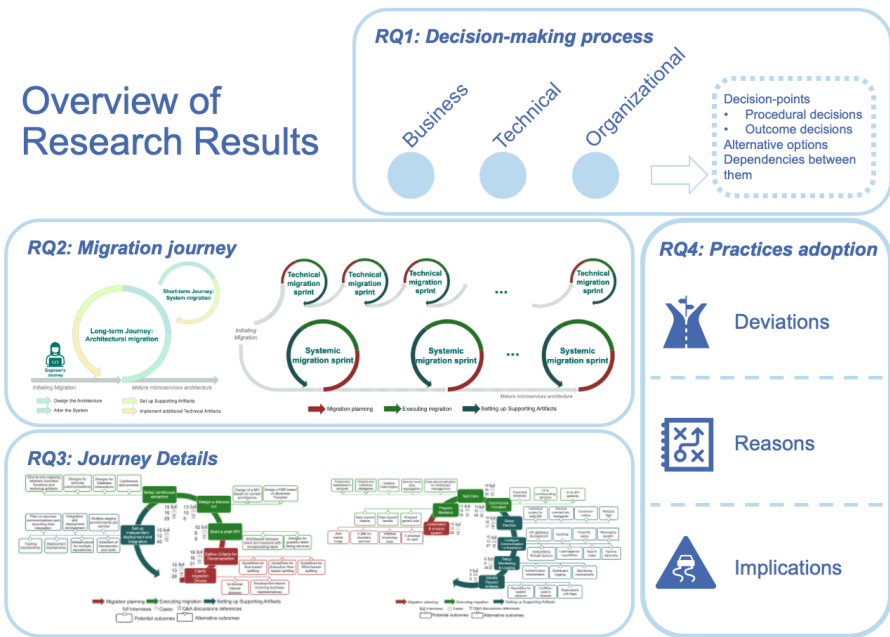

Figure 1.4: Overview of contributions per research question

1.6.1 Contribution 1: Decision-making process

The first contribution of this thesis is the proposed decision-making process that covers comprehensively the multidimentionality 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 on 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, a mechanism was devised in paper A to uncover implicit decisions that software engineers make. There were three predominant ways in identifying decisions in our interview material. First, when interviewees mentioned that they had to make a decision from different alternatives. The second way was when interviewees seemed unsure about a choice the had made and discussed the rationale behind it or choose two options simultaneously. The third way was when we identified different courses of actions 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 are evidence of decisions influencing decisions of other dimensions.

In paper A, the decision-making processes that happens on all levels of a microservices migration project is 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. Furthermore, this contribution helps in understanding the impact of microservices on organizational aspects (structure, processes, VPs etc.) and demonstrate the eventual nature of migrations in organizations. A strong emphasis is given on 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, being the business, technical and organizational dimension.

In the business dimension, the developed theory reports the need to first create engagement across the organization for migrating to a MSA. Specifically, engineers that 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 are feeding 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 dimensions 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 dimention 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 on 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.

1.6.2 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. The contribution of these papers' results is that they show that migration projects are continuous improvement initiatives instead of one-of projects. Both paper B and C organize different aspects that describe past migrations and present them in a continuous endeavour that takes place in iterations. Understanding the progress of migration projects in the aggregated process can help engineers in having awareness of the progress of different modes of change. Also, both paper B and paper C showcase the different pace in different levels of the migration.

For example, paper B presents two main modes of the work during a migration that demonstrate different modes of change. 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.

Paper C builds on that work and with the analysis of an additional dataset confirms the initial theory and modifies it accordingly. Specifically, the aggregated migration journeys of 16 organizations is complemented through the analysis of 215 posts from StackOverflow, that discuss microservices migrations. Paper C uses the StackExchange data explorer [2] 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 to a different

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

distinction between the phases, resulting to merging two of them. In paper C, 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 that the development and operations are configured in order to support effectively the new paradigm that microservices bring.

1.6.3 Contribution 3: The constituent elements of the 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 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 are explaining in detail specific activities that take place during migration iterations. Moreover, paper C 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 a 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 is concerning 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 for the grand scheme of things, but it is still crucial in mirating 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 in paper C 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 in order 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 shows how activities of migrations connect to each other and how they materialize to solution outcomes. Such findings contribute to forming abstract patterns into actionable practical

activities with concrete solution outcomes.

1.6.4 Contribution 4: The organizational aspect in software engineering

Finally, this thesis contributes with a detailed view on organizational aspects in software engineering. Specifically, through investigating the adoption of practices as well as the organizational implications of software architecture change. Paper A and C include suggested changes on the software development methodology of migrating teams. However, before investigating such changes, it is critical to understand how software engineers react to given guidelines (e.g., Scrum). Specifically, development methodologies can be altered with time and deviations from guidelines can arise.

Paper D empirically evaluates how software engineering teams adopt Scrum in their software development process. Therefore, paper D presents how teams deviate from best practice, why these deviations happen and what the consequences or implications of these deviations are. The identified reasons that the thirteen listed deviations showed are grouped into human factors, organizational structures and complexity of the teams' work. Furthermore, the identified implications are on the product development process and on the teamwork of the teams deviating. An important contribution of paper D is the attempt to understanding rather than judging deviations from guidelines.

This understanding, can be transitioned into how we expect organizations to migrate towards a MSA. Hence, we can start observing the organizational aspects in microservices migrations, especially when they are not entirely aligned with technical aspects. Paper A demonstrates how organizational change is a key accompanying dimension to all the changes that happen during a technical migration. Specifically, a substantial part of the decision-making is regarding organizational aspects of software engineering and the analyzed interviews revealed how important these aspects are. Finally, both paper B and paper C touch upon the necessary preparations of the organization to accept the change that such migrations entail.

Moreover, deviations can indicate the need for process tailoring and this can be considered when applying processes for decomposing systems to microservices. Instead of unintentionally deviating from an operating model, it is worth considering to intentionally structure the operating model on the team and deliver good software by design rather by coincidence.

1.7 Discussion

This thesis gives a strong emphasis in microservices migrations as a socio-technical endeavor. Paper A, paper B and paper C do 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 [18]. Furthermore, as the outcomes from analyzing microservices migrations resulted in processes and taxonomies, the thesis demonstrates a high consideration of developers perspectives in adopting processes and guidelines through paper D.

1.7.1 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 [53]. 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 [54]. 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 iterative nature of engineering software [55]. Therefore, current and future research needs to establish more dynamic ways of explaining software architecture, that showcases the evolution that systems go through.

MSA migrations entail decisions and communication from engineers all the way up to executives. Software architecture is the representation of systems in different levels of abstraction, but also the actual implementation. Architectural representations of a system are the designs or the shared formal understanding between stakeholders. Such representations can be used to navigate systems in detail, but also have an overview of the big picture [53]. Papers B and C build on these ideas and draw empirical results along the lines of propagating the change on different levels of detail and at the same time link narrow technical details with broad organizational structures.

Paper C goes even further on these ideas of existing literature and demonstrates the different activities of the development of the architecture when migrating. Software architectures are produced by a disciplined approach to designing and architecting parts of a system. Coding and programming entails most of the designing part in software development. Therefore, one can argue that software development is also architecture development [54]. There are arguments that software architecture is design on a diffferent, bigger scale. However, paper C showcases that there are things that matter at all levels of detail and levels of abstractions. There are small details that influence large, systemic design choices and vice versa - grand design choices that influence

small, technical details. 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.

1.7.2 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 [10, 20, 23, 24]. This diversity indicates the complexity that exists when migrating towards a MSA, as do individual studies that investigate instances of such migrations [17]. 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 paper 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.

Migration drivers, business and organizational needs, as well as 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 [3]. Clarifying migration drivers complements current research that describes the benefits of migrating towards microservices, being technical or economic [37, 56], 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 [9]. 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.

The different aspects discussed in this thesis are in line with socio-technical systems design and this connection is visible across this thesis in different forms. The established process, the organizational structure and the technical infrastructure, along with the inherited human-driven complexity in their relations [18, 45] is taken into account in all appended studies. 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 paper A, B and C. On the other hand, the development methodologies are also altered

with time and deviations from guidelines can arise, as presented in paper D. Hence, this research consists of a critical view on when it is possible and useful to migrate and until which level of migration completion.

1.7.3 The decision-making part of 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 [54]. A decision-making process is therefore important as also indicated by other studies [21] and paper A investigates decision-making from many different organizations. Decision-making of software engineers takes place in an individual level [42], in groups/teams or small software organizations [44] and in large organizations that develop software [57]. 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. [20], 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 [21].

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.

1.7.4 The implications of the developed theories 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 [11]. 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 [19, 24]), designing service cuts (relating to services designs [39,58]) and continuously re-extracting services (relating to designing MSAs [21]). 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.

1.7.5 Adopting processes and guidelines

Process designers, in both research and practice, need to reflect on adoption and deviations when creating or maintaining a migration process. In this thesis, it is investigated how development practices of software engineers evolve over time, in order to keep in careful consideration the reaction of engineers towards processes and guidelines. The objective of this work is to indicate how process guidelines for MSA migrations and software development methodologies are propagated into activities of engineers. To achieve this, paper A approaches change on one hand from the perspective of intended deviations on the operational model of an organization, through explicit decision-making. On the other hand, paper D approaches change from the perspective of unintended deviations from existing guidelines that were supposed to be followed. Specifically, the empirical research conducted in paper D derives inductively from engineers' experiences the adoption and deviations of the scrum framework and what are the reasons and implications of deviating from the guidelines.

Therefore, based on the findings of paper D we can derive that process designers cannot assume that developers are going to follow a process by the guide. It is rather more accurate to assume that developers will deviate and this is aligned with existing research [26, 29]. Hence, the design approach of the process can be changed accordingly and deviations can indicate the need for process tailoring [31]. In paper D an approach is described on how to deviate by design and this approach can be used potentially also on the migration processes presented in papers A, B and C.

Moreover, process improvement and process design from both, research and practice should take into account both intentional and unintentional change. As software evolves, there can be a difference between the representation of a system and the actual implementation. Hence, it can be argued that the architecture is essentially what is actually implemented (even though it deviates from the software representation). This thesis discusses intended, planned change as well as unintended, coincidential change of processes. Specifically, paper A and paper D present an opposing critical view on change, with paper A discussing intended change and paper D discussing unintended change. There are many factors that can lead to change, being a migration or a methodology change [28]. Hence, the findings of this thesis reflect on the needed consideration of different aspects for tailoring processes to the requirements of engineers [31]. It is beneficial to consider the unintentional deviations from an operating model when requiring to intentionally re-structuring the operating model on the team. Also, it is beneficial during process execution to be aware of unintentional changes that engineers might make on the process for different reasons.

1.8 Conclusion

As software systems grow large, both in size and complexity, it becomes difficult to update them. While modernizing a large or growing software system, it is becoming popular to aim on implementing a MSA. 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. 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. First of all, 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 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 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. Also, the aim is to understand how the adoption of Scrum evolves over time and what the reasons and implications of deviating from such methodologies are.

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. Thirdly, this thesis investigates the organizational aspect of software development and sheds light on how development practices of software engineers deviate from the intended practices and guidelines.

1.9 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 machine learning applications is integrated and deployed to existing systems that operate on a large scale. In addition, next steps include to investigate the evolution of specific elements of the software architecture. For example, in future work it is intended to investigate how testing and the testing architecture 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.

Bibliography

[1] A. Brand, L. Allen, M. Altman, M. Hlava, and J. Scott, "Beyond authorship: attribution, contribution, collaboration, and credit," *Learned Publishing*, vol. 28, no. 2, pp. 151–155, 2015.

[2] S. Ducasse and D. Pollet, "Software architecture reconstruction: A process-oriented taxonomy," *IEEE Transactions on Software Engineering*, vol. 35, no. 4, pp. 573–591, 2009.

[3] J. Bosch, "Speed, data, and ecosystems: The future of software engineering," *IEEE Software*, vol. 33, no. 1, pp. 82–88, 2016.

[4] M. F. Gholami, F. Daneshgar, G. Beydoun, and F. Rabhi, "Challenges in migrating legacy software systems to the cloud — an empirical study," *Information Systems*, vol. 67, pp. 100–113, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306437917301564

[5] Z. Cai, L. Zhao, X. Wang, X. Yang, J. Qin, and K. Yin, "A pattern-based code transformation approach for cloud application migration," in *2015 IEEE 8th International Conference on Cloud Computing*, 2015, pp. 33–40.

[6] J. Thönes, "Microservices," *IEEE software*, vol. 32, no. 1, pp. 116–116, 2015.

[7] U. Zdun, E. Wittern, and P. Leitner, "Emerging Trends, Challenges, and Experiences in DevOps and Microservice APIs," *IEEE Software*, vol. 37, no. 1, pp. 87–91, jan 2020.

[8] N. Dragoni, I. Lanese, S. T. Larsen, M. Mazzara, R. Mustafin, and L. Safina, "Microservices: How to make your application scale," in *Perspectives of System Informatics*, A. K. Petrenko and A. Voronkov, Eds. Cham: Springer International Publishing, 2018, pp. 95–104.

[9] O. Zimmermann, "Microservices tenets: Agile approach to service development and deployment," *Computer Science - Research and Development*, vol. 32, no. 3-4, pp. 301–310, jul 2017.

[10] P. Di Francesco, P. Lago, and I. Malavolta, "Architecting with microservices: A systematic mapping study," *Journal of Systems and Software*, vol. 150, pp. 77–97, 2019.

[11] S. Newman, *Building microservices: designing fine-grained systems.* " O'Reilly Media, Inc.", 2015.

[12] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, *Microservices: Yesterday, Today, and Tomorrow*. Cham: Springer International Publishing, 2017, pp. 195–216.

[13] J. Fritzsch, J. Bogner, S. Wagner, and A. Zimmermann, "Microservices migration in industry: Intentions, strategies, and challenges," in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2019, pp. 481–490.

[14] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices architecture enables devops: Migration to a cloud-native architecture," *IEEE Software*, vol. 33, no. 3, pp. 42–52, 2016.

[15] J. Soldani, D. A. Tamburri, and W.-J. Van Den Heuvel, "The pains and gains of microservices: A systematic grey literature review," *Journal of Systems and Software*, vol. 146, pp. 215–232, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121218302139

[16] R. Khadka, A. Saeidi, S. Jansen, and J. Hage, "A structured legacy to soa migration process and its evaluation in practice," in *2013 IEEE 7th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems*, 2013, pp. 2–11.

[17] D. Taibi, V. Lenarduzzi, and C. Pahl, "Microservices Anti-patterns: A Taxonomy," in *Microservices*. Springer International Publishing, 2020, pp. 111–128.

[18] G. Baxter and I. Sommerville, "Socio-technical systems: From design methods to systems engineering," *Interacting with Computers*, vol. 23, no. 1, pp. 4–17, 2011.

[19] H. Knoche and W. Hasselbring, "Using microservices for legacy software modernization," *IEEE Software*, vol. 35, no. 3, pp. 44–49, 2018.

[20] S. Hassan, R. Bahsoon, and R. Kazman, "Microservice transition and its granularity problem: A systematic mapping study," *Software - Practice and Experience*, vol. 50, no. 9, pp. 1651–1681, 2020.

[21] M. Waseem, P. Liang, G. Márquez, M. Shahin, A. A. Khan, and A. Ahmad, "A decision model for selecting patterns and strategies to decompose applications into microservices," in *Service-Oriented Computing*, H. Hacid, O. Kao, M. Mecella, N. Moha, and H.-y. Paik, Eds. Cham: Springer International Publishing, 2021, pp. 850–858.

[22] J. Fritzsch, J. Bogner, A. Zimmermann, and S. Wagner, "From monolith to microservices: A classification of refactoring approaches," in *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*. Springer, 2018, pp. 128–141.

[23] M. Waseem, P. Liang, M. Shahin, A. Di Salle, and G. Márquez, "Design, monitoring, and testing of microservices systems: The practitioners' perspective," *Journal of Systems and Software*, vol. 182, p. 111061, 2021. [Online]. Available: https://doi.org/10.1016/j.jss.2021.111061

[24] S. Newman, *Monolith to microservices: evolutionary patterns to transform your monolith*. O'Reilly Media, 2019.

[25] M. Waseem, P. Liang, and M. Shahin, "A Systematic Mapping Study on Microservices Architecture in DevOps," *Journal of Systems and Software*, vol. 170, p. 110798, 2020. [Online]. Available: https://doi.org/10.1016/j.jss.2020.110798

[26] P. Clarke and R. V. O'Connor, "The situational factors that affect the software development process: Towards a comprehensive reference framework," *Information and Software Technology*, vol. 54, no. 5, pp. 433–447, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.infsof.2011.12.003

[27] M. Unterkalmsteiner, T. Gorschek, A. K. Islam, C. K. Cheng, R. B. Permadi, and R. Feldt, "Evaluation and measurement of software process improvement-A systematic literature review," pp. 398–424, 2012.

[28] M. R. Lazwanthi, A. Alsadoon, P. W. Prasad, S. Sager, and A. Elchouemi, "Cultural impact on agile projects: Universal agile culture model (UACM)," in *2016 7th International Conference on Information and Communication Systems, ICICS 2016*. Institute of Electrical and Electronics Engineers Inc., may 2016, pp. 292–297.

[29] M. A. A. Da Silva, R. Bendraou, J. Robin, and X. Blanc, "Flexible deviation handling during software process enactment," in *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOC*. Institute of Electrical and Electronics Engineers Inc., 2011, pp. 34–41.

[30] J. Klunder, R. Hebig, P. Tell, M. Kuhrmann, J. Nakatumba-Nabende, R. Heldal, S. Krusche, M. Fazal-Baqaie, M. Felderer, M. F. Genero Bocco, S. Kupper, S. A. Licorish, G. Lopez, F. McCaffery, O. Ozcan Top, C. R. Prause, R. Prikladnicki, E. Tuzun, D. Pfahl, K. Schneider, and S. G. MacDonell, "Catching up with Method and Process Practice: An Industry-Informed Baseline for Researchers," in *Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP 2019*. Institute of Electrical and Electronics Engineers Inc., may 2019, pp. 255–264.

[31] G. Kalus and M. Kuhrmann, "Criteria for software process tailoring: A systematic review," in *ACM International Conference Proceeding Series*, 2013, pp. 171–180.

[32] V. Lenarduzzi, F. Lomio, N. Saarimäki, and D. Taibi, "Does migrating a monolithic system to microservices decrease the technical debt?" nov 2020.

[33] N. Venkatraman and V. Ramanujam, "Measurement of business performance in strategy research: A comparison of approaches," *Academy of Management Review*, vol. 11, no. 4, pp. 801–814, 1986. [Online]. Available: https://doi.org/10.5465/amr.1986.4283976

[34] P. Jamshidi, C. Pahl, N. C. Mendonça, J. Lewis, and S. Tilkov, "Microservices: The journey so far and challenges ahead," *IEEE Software*, vol. 35, no. 3, pp. 24–35, 2018.

[35] P. Di Francesco, P. Lago, and I. Malavolta, "Migrating towards microservice architectures: An industrial survey," in *2018 IEEE International Conference on Software Architecture (ICSA)*, 2018, pp. 29–2909.

[36] T. Cerny, M. J. Donahoo, and J. Pechanec, "Disambiguation and comparison of SOA, microservices and self-contained systems," *Proceedings of the 2017 Research in Adaptive and Convergent Systems, RACS 2017*, vol. 2017-Janua, no. 4, pp. 228–235, 2017.

[37] D. Taibi, V. Lenarduzzi, and C. Pahl, "Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation," *IEEE Cloud Computing*, vol. 4, no. 5, pp. 22–32, 2017.

[38] X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, W. Li, and D. Ding, "Fault Analysis and Debugging of Microservice Systems: Industrial Survey, Benchmark System, and Empirical Study," *IEEE Transactions on Software Engineering*, vol. 47, no. 2, pp. 243–260, feb 2021.

[39] M. Gysel, L. Kölbener, W. Giersche, and O. Zimmermann, "Service cutter: A systematic approach to service decomposition," in *Service-Oriented and Cloud Computing*, M. Aiello, E. B. Johnsen, S. Dustdar, and I. Georgievski, Eds. Cham: Springer International Publishing, 2016, pp. 185–200.

[40] G. Mazlami, J. Cito, and P. Leitner, "Extraction of Microservices from Monolithic Software Architectures," in *Proceedings - 2017 IEEE 24th International Conference on Web Services, ICWS 2017*. Institute of Electrical and Electronics Engineers Inc., sep 2017, pp. 524–531.

[41] M. Camilli and B. Russo, "Modeling performance of microservices systems with growth theory," *Empirical Software Engineering*, vol. 27, no. 2, pp. 1–44, 2022.

[42] H. van Vliet and A. Tang, "Decision making in software architecture," *Journal of Systems and Software*, vol. 117, pp. 638–644, 2016.

[43] C. Zannier, M. Chiasson, and F. Maurer, "A model of design decision making based on empirical results of interviews with software designers," *Information and Software Technology*, vol. 49, no. 6, pp. 637–653, 2007.

[44] S. Rekha V and H. Muccini, "Group decision-making in software architecture: A study on industrial practices," *Information and Software Technology*, vol. 101, pp. 51–63, sep 2018. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0950584918300740

[45] P. Lenberg, R. Feldt, and L. G. Wallgren, "Behavioral software engineering: A definition and systematic literature review," *Journal of Systems and Software*, vol. 107, pp. 15–37, 2015.

[46] A. Tang, M. Razavian, B. Paech, and T.-M. Hesse, "Human aspects in software architecture decision making: a literature review," in *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2017, pp. 107–116.

[47] K. J. Stol, P. Ralph, and B. Fitzgerald, "Grounded theory in software engineering research: A critical review and guidelines," *Proceedings - International Conference on Software Engineering*, vol. 14-22-May-2016, no. Aug 2015, pp. 120–131, 2016.

[48] K. Charmaz, *Constructing grounded theory*. sage, 2014.

[49] S. Baltes and S. Diehl, "Usage and attribution of stack overflow code snippets in github projects," *Empirical Softw. Engg.*, vol. 24, no. 3, p. 1259–1295, jun 2019. [Online]. Available: https://doi.org/10.1007/s10664-018-9650-5

[50] A. Tahir, J. Dietrich, S. Counsell, S. Licorish, and A. Yamashita, "A large scale study on how developers discuss code smells and anti-pattern in stack exchange sites," *Information and Software Technology*, vol. 125, p. 106333, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584920300926

[51] A. Abdellatif, D. Costa, K. Badran, R. Abdalkareem, and E. Shihab, "Challenges in Chatbot Development: A Study of Stack Overflow Posts," *Proceedings - 2020 IEEE/ACM 17th International Conference on Mining Software Repositories, MSR 2020*, pp. 174–185, 2020.

[52] T. Lopez, T. Tun, A. Bandara, L. Mark, B. Nuseibeh, and H. Sharp, "An anatomy of security conversations in stack overflow," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, 2019, pp. 31–40.

[53] P. Kruchten, "The 4+1 view model of architecture," *IEEE Software*, vol. 12, no. 6, pp. 42–50, 1995.

[54] J. Bosch, *Design and use of software architectures: adopting and evolving a product-line approach*. Pearson Education, 2000.

[55] P. Kruchten, H. Obbink, and J. Stafford, "The past, present, and future for software architecture," *IEEE Software*, vol. 23, no. 2, pp. 22–30, 2006.

[56] A. Singleton, "The economics of microservices," *IEEE Cloud Computing*, vol. 3, no. 5, pp. 16–20, 2016.

[57] Z. Li, P. Liang, and P. Avgeriou, "Architectural Technical Debt Identification Based on Architecture Decisions and Change Scenarios," *Proceedings - 12th Working IEEE/IFIP Conference on Software Architecture, WICSA 2015*, no. 895528, pp. 65–74, 2015.

[58] D. Taibi and V. Lenarduzzi, "On the Definition of Microservice Bad Smells," *IEEE Software*, vol. 35, no. 3, pp. 56–62, may 2018.