Facing the Giant: a Grounded Theory Study of Decision-Making in Microservices Migrations

Hamdy Michael Ayas
ayas@chalmers.se
CSE Department
Chalmers | University of Gothenburg
Gothenburg, Sweden

Philipp Leitner
philipp.leitner@chalmers.se
CSE Department
Chalmers | University of Gothenburg
Gothenburg, Sweden

Regina Hebig
hebig@chalmers.se
CSE Department
Chalmers | University of Gothenburg
Gothenburg, Sweden

ABSTRACT

Microservices migrations are challenging and expensive projects with many decisions that need to be made in a multitude of dimensions. Existing research tends to focus on technical issues and decisions (e.g., how to split services). Equally important organizational or business issues and their relations with technical aspects often remain out of scope or on a high level of abstraction. The objective of this study is to holistically chart the decision-making that happens on all dimensions of a migration project towards microservices. We investigate 16 migration cases, by conducting a grounded theory interview study with 19 participants that recently underwent a migration. We also provide an initial validation via a Web-based survey with 52 respondents. Our study approaches the topic with a strong focus on the human aspect of a migration, through stakeholders, their concerns and the decisions they need to make as part of the migration. We identify 3 decision-making processes consisting of 22 decision-points in total, and their typical alternatives or options. The decision-points are related to creating stakeholder engagement and assessing feasibility, technical implementation, and organizational restructuring. Our study provides an initial theory of decision-making in migrations to microservices, and outfits practitioners with a roadmap of which decisions they should be prepared to make and at which point in the migration.

CCS CONCEPTS

• **Software and its engineering** → *Cloud computing*; *Software design engineering.*

KEYWORDS

microservices, migration, decision-making, grounded theory

1 INTRODUCTION

Organizations in many industries are increasingly adopting microservices technologies to design, develop, test and maintain their software systems. Development with microservices can happen in new software systems, but more commonly an existing system needs to be migrated to a microservices-based architecture (MSA) [5]. Therefore, migrations towards microservices are increasingly gaining popularity in both industry and academia [17]. Also, early research and documented work on microservices demonstrates the complexity in the nature of designing and developing microservices [24, 33]. Hence, migrating towards microservices is also a complex endeavour. Migrations have many activities and challenges that need to be identified and investigated systematically [10]. Finally, architectural migrations are naturally heavy in decision-making. Defining the decision-making processes of such initiatives helps to better understand them and improve their realization [2].

Migrating towards microservices can be rewarding for organizations [32], but it also contains many decisions to make along the way [8]. Consequently, it is not surprising that substantial previous research has investigated questions surrounding microservices migrations [5, 14]. These works tend to focus on technical issues and decisions (e.g., how to split services [16, 26]) and on specific questions of software architecture [29]. Also, there are many studies on how to technically enact the migration (e.g., through program transformation or service identification [15]). However, the solutions proposed are often not sufficiently satisfying in supporting engineers and their decision-making during migrations [8]. Furthermore, microservices migrations are transformative on organizations as a whole, and the non-technical aspects of migrations (e.g., how to assess the business case for the migration or how to restructure teams) are less well-understood than the technical aspects. Research and best practices stemming from industry provide quite comprehensive approaches on migrations, covering many aspects [3, 25]. However, there is a gap on comprehensive approaches that are on an abstraction level closer to the operational choices that organizations make during migrations. Also, there is a need in empirically understanding the details of migrations from engineers' point of view.

Hence, the objective of this study is to holistically chart the decision-making processes that happen on all levels of a microservices migration project, inductively from empirical evidence. 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. Our study approaches the topic with a strong focus on the human aspect of a migration, through stakeholders, their concerns and the decisions they need to make as part of the migration. Hence, in this study we analyze 16 different cases of migrations towards microservices from 16 different organizations, via conducting a grounded theory based interview study with 19 developers. All developers have recently been part of a migration towards microservices. In addition, we conduct a partial triangulation and validation through a Web-based survey with 52 responses.
Specifically, we study the following research questions:

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

> ***RQ1.1****: What are the decisions that organizations make during a migration towards microservices?*

> ***RQ1.2***: *At what point in the migration are these decisions made?*

RQ2: *What are typical options that organizations can choose in each of these decisions?*

We construct an initial theory of decision-making in an organizational level, during migrations towards microservices. The decision-making processes we construct consist of decision-points, typical options, and dependencies between them (e.g., follow-up decisions based on the outcome of a previous decision). We identify 22 decision-points with their potential options. These relate to the assessment of technical feasibility, the validation of the migration through a business case, the technical implementation, as well as the restructuring of the organization and its operations. Moreover, we distinguish two types of decision-points: (1) procedural decision-points (decisions about how to continue the migration project, i.e., which strategies to use), and (2) outcome decision-points (decisions about the MSA that shall result from the migration).

Through the identified decision-making processes, we demonstrate the choices that organizations make during the course of a migration, from its early stages. The contents of our identified decision-making processes first complement existing knowledge on the benefits of microservices. We demonstrate that microservices are not only motivated by economic gains of efficiency, but also by gains in effectively and continuously delivering business value. Subsequently, we demonstrate a comprehensive view of migrations with decision-making not only on the technical dimension (e.g. developing microservices), but also the organizational (e.g. managing teams). Our focus is mainly on understanding the decisions that stakeholders (e.g. developers, managers) make during migrations and the situations they come across. We do not argue that we provide a single source of truth on how to migrate. In fact, we believe that such a thing does not exist due to the complexity and socio-technical nature of organizations migrating.

2 RELATED WORK

With migrating to microservices, organizations aim to deconstruct their systems into smaller, independent services [12]. These services need to be in principle decoupled from each other, with minimal dependencies [24]. Also, microservices are individually owned from responsible teams that design them around business domains [33]. To achieve the benefits of microservices, there are design or architectural patterns developed [30]. These patterns include organizing software systems around business capabilities, enabling automated deployment, facilitating intelligence in the endpoints and decentralizing the control of programming languages and data [11]. Therefore, usually there is a large leap between a monolith and a microservices architecture and a migration / transition is the crucial project that takes the organization through the leap [17].

Microservices enable many benefits in the development and operations of software systems [32]. For example, the scalability potentials of microservices are unprecedented due to the possibility of dynamically scaling-up and scaling-down parts of an application [13]. Furthermore, the modular organization of systems with minimal dependencies, may offer improved maintainability [32] and organizational agility [33]. Benefits in agility and maintainability are possible in novel ways due to the independence and flexibility of microservices development [4]. Also, characteristics of such an architecture (e.g., infrastructure automation) enable continuous delivery, improving operational efficiency [27]. An efficiency example is lowering the average size of development teams in comparison to monolith systems, and reducing domain-specific redundancy between microservices [23]. Hence, the proposition of migrating to microservices enables substantial economic potential in efficiently developing and managing complex software systems [27]. However, the aspect of organizational and business effectiveness is equally important [31], even though it is not studied as extensively in the context of microservices.

Organizations are increasingly migrating their software architectures by adopting microservices and researchers are increasingly investigating solutions for such transitions [17]. A substantial amount of previous research has investigated the area surrounding microservices migrations and their characteristics [5, 14, 29]. In addition, there are solutions researched on how to technically enact the migration. These solutions include splitting a system, transforming the code of an application or identifying services [15]. These solutions often provide tools on how to identify and decompose or extract services, assuming a technical viewpoint on the migration [16]. This is not always ideal, as other aspects are neglected that usually come along with a migration, related to managing the entire change of an organization [25]. Industry-based research uses influential work from the discipline of change management to demonstrate how to lead the change of a migration. For example, with the 8-steps model to transforming an organization [19]. Hence, it could prove valuable to go even further in detail in such influential work, for example on specific activities on how to manage resistance to change through education, participation, facilitation, negotiation and coercion [20]. This will help to understand the ways for consistently achieving the strategic alignment needed to leverage the technology [18] of microservices in this case.

Migrations are intensive in operational decisions [8]. Just like every transformation initiative, migrating to microservices entails many risks to consider [22]. For example, starting from a greenfield to migrate can be very expensive but re-factoring an established system is a long-lasting endeavour, influenced by a broad range of aspects [15]. In such cases, understanding the decision-making process is beneficial [2]. Decision-making can be incorporated in all architectural views to facilitate our understanding of systems which is particularly valuable during change [21]. Hence, it is not only needed to understand the criteria for decision-making in migrations towards microservices [8], but also to understand the decision-making processes (with their other constituent elements).

3 METHODOLOGY

Our primary research method was a grounded theory based interview study with practitioners who have recently participated in a microservice migration project. Additionally, we conducted a triangulation and validation step using a Web-based survey. Interview guide and survey materials can be found in our replication package [1]. We omit interview transcripts from the replication package to preserve interviewee privacy and protect potential commercial interests of our interviewee's employers.

3.1 Interviews

In the interview step, we relied on techniques found in Grounded Theory (GT) [9], namely coding, memoing, sorting, constant comparison and theoretical saturation. Based on guidelines for GT in software engineering research, we cannot claim to use the classic GT method. Instead, we used constructivist GT as we had significant previous exposure to literature prior to the study [28]. Such that some of our themes align with both, previous research [5, 17] as well as common practitioner guidance on best practices for microservices migration [25]. When conducting the interviews, we used a semi-structured interview guide, which we constructed based on our research questions. However, we gave participants significant freedom to describe their own migration journeys in their own words.

Participants: Due to the well-known challenges of recruiting a representative sample of software developers for interview studies, we had to rely on purposive sampling [6] and our personal network (e.g., through current and previous projects, colleagues, or students). Further, we used a snowballing approach, where we asked each interviewee to refer us further to other potential participants. We used a saturation approach [9] where we continued inviting participants in parallel to data analysis as long as new insights were gained. Our acceptance criteria for interviewees were *(a)* software professionals (not students) who *(b)* have participated in or have closely observed a microservices migration project within their professional work. An overview of the participants is found in Table 1. We have interviewed 19 professionals from 6 different countries (Cyprus, UAE, Germany, Romania, Sweden, The Netherlands), of which 18 were male and one female. Interviewers had on average 7.5 years of experience (ranging from 2 to 21) and they have worked at medium to large companies in twelve business domains. In addition, the migration cases are about systems delivered to external customers (e.g. Enterprise SaaS), in-house enterprise solutions for internal users and also Software Applications sold as a service (e.g. mobile app).

Case	*Interview*	*Role*	*Experience*	*Industry*
C1	I1	Full stack developer	2 (1)	Enterprise Software
C2	I2	Software Engineer	2 (2)	Gaming
C1	I3	Senior Team Leader	12 (2)	Enterprise Software
C3	I4	Software Engineer	2 (1)	Banking Software
C4	I5	Software Engineer	19 (1.5)	Banking Software
C1	I6	Software Engineer	2 (1)	Enterprise Software
C5	I7	Software Engineer	7 (2)	Aviation Software
C6	I8	Software Developer	3 (3)	Telecommunications
C7	I9	Computer Scientist	5 (5)	Enterprise Software
C8	I10	Principal Software Engineer	7 (4)	Cloud Computing
C9	I11	Software Engineer	6 (3)	Marketing Analytics
C10	I12	Data Engineer	6 (2)	Healthcare Software
C11	I13	Senior Cloud Architect	10 (5)	Cloud Computing
C12	I14	Software Engineer	4 (1.5)	Energy Software
C12	I15	Software Architect	4 (4)	Energy Software
C13	I17	Co-founder	8 (5)	Logistics / Planning
C14	I16	Software Architecture Consultant	13 (4)	Logistics / Planning
C15	I18	CTO	10 (6)	Manufacturing
C16	I19	Enterprise Architect	21 (5)	Manufacturing

Table 1: Participants of interview study. Experience is reported in years, values in brackets are years of microservices experience.

Protocol: We conducted our interviews over a period of six months. Each interview took between 30 and 60 minutes. Initial interviews were conducted face-to-face, but due to the ongoing COVID-19 pandemic as well as geographical distance most of our interviews had to be carried out through video conferencing. Prior to each interview, participants were asked to sign a consent form, and consent to recording the interview. Further, participants were made aware that they can drop out of the study at any point, which no interviewee made use of. We did not offer financial rewards to study participants.

Analysis: In an ongoing process parallel to data collection, we performed initial, focused and theoretical coding based on the constructivist variant of GT [9, 28]. In initial coding we fractured the data to find relevant statements. In focused coding, we aggregated and connected those excerpts into categories and themes until achieving saturation. In theoretical coding we specified the relationships of the connected categories and integrated them into a cohesive theory. Initial coding was conducted by the first author. All three authors collaborated in focused coding in three card sorting and memoing sessions lasting three to four hours each. All resulting findings are supported by statements from multiple participants.

3.2 Web-Based Survey

Additionally, we conducted a second study step using a Web-based survey. The survey was targeting practitioners with interest (but not necessarily experience) in microservices migrations, and entailed in total 28 closed questions. Questions were designed after the analysis of interview data was completed to validate main interview outcomes as well as collect additional data for questions that could not be directly answered based on the first study step. We used the survey tool Typeform[1] to implement the survey. Despite a limited number of respondents (52 complete responses), we argue that our survey adds to our interview findings by providing a first limited-scale quantitative angle to our study.

We advertised our survey over social media (e.g. through LinkedIn) and among our industrial contacts (including participants in the interview study step). Survey responses were entirely anonymous and participation was voluntary, without offering rewards for completion (neither monetary nor otherwise). Our survey had a completion rate of 31.8% and took on average 11 minutes to complete.

3.3 Threats to Validity

We designed out research as a mixed-method study to be able to triangulate interview results through quantitative survey data, allowing us to mitigate some common threats inherent in interview studies. However, some threats that are inherent in our chosen study methodology remain, which readers should take into consideration.

External Validity. For both study steps, we cannot claim representativeness of our study demographics for the software industry in general, as both populations have been sampled through our personal network and using a voluntary, opt-in procedure. To mitigate this threat, we selected interview participants to cover companies of different sizes, in different domains, and in different geographical regions. In the survey we avoided collecting detailed company or

[1]http://typeform.com/

geographical information to prevent de-anonymising participants. Further, a voluntary survey design is always susceptible to self-selection bias: respondents uninterested in the field of study are unlikely to participate in a survey such as ours.

Internal Validity. In terms of internal validity, a threat is that we were, through our previous interest in the field, pre-exposed to existing research as well as the extensive practitioner-focused guidance on how to conduct microservice migrations. This may have biased our interview design, and may have led that some decision-points (e.g., those not discussed in earlier work) may have been given less prominence or judged as unimportant during analysis. In general, a limitation of our study design is that we cannot claim that the identified list of decision-points and alternatives is necessarily complete.

4 RESULTS

Our study shows that migrations towards microservices entail a multitude of decisions on different dimensions and of different types. We broadly distinguish between the *decision-making process of creating engagement*, which then influences the *decision-making process on the technical dimension* and the *decision-making process on the organizational dimension.* On all dimensions, practitioners encounter two types of decisions that take place in certain points of time: *procedural decision-points* (i.e., points during the migration where they need to decide how to continue the migration process) and *outcome decisions-points* (e.g., points during the migration where the team needs to agree on specific architectures or technologies as outcome of the migration process). Both types of decision-points are only infrequently restricting practitioners to make a commitment to a single approach or technology — instead, we observe that practitioners often choose to follow multiple complementary approaches.

We now discuss decisions on creating engagement as well as organizational and technical decisions. We also discuss their dependencies and implications for migration projects, in more detail. We generally focus on the "flow" of decision-points and common options. Note that providing an exhaustive list of all possible options for every decisions is neither feasible nor the scope of this article.

4.1 Decisions on Creating Engagement

Decisions on creating engagement typically relate to whether a migration is an endeavour that the organization wants to pursue in the first place. This requires evaluating technical feasibility as well as creating engagement with all key stakeholders. We identify key stakeholders as top management, middle management and operational personnel. Top management's buy-in is essential, as they act as the funding agency of the migration. Middle management is commonly the source of knowledge about how cost reductions or profit increases can be enacted in practice. Finally, operational personnel (also including software developers and architects) engineer the new architecture, and are responsible for the development and delivery of the system.

We observe that the studied cases of migration projects include decisions that address the concerns of all three of these stakeholder groups. Firstly, technical feasibility is evaluated (predominantly to convince the operational and middle management stakeholders). Then follows the construction of convincing business cases for top management. Therefore, creating management buy-in predominantly entails establishing a business case describing how microservices lead to reduced costs and/or increased profits (e.g. from a better product or service) for a strategic (and often very expensive) migration project. It should be noted that this process of creating and evaluating buy-in is not a one-time procedure – instead, successful migration projects continuously evaluate and re-evaluate if microservices are still the right fit for their project or company.

> "*the microservice approach allows an iterative growth [...] we start [...] a very shallow data inventory, then I tried to add applications on this shallow data inventory, then I tried to drive the customer to a point where they say 'Okay' for the next application*" -I18

4.1.1 Technical Feasibility and Exploration of Opportunities. This first phase of the migration (depicted in Figure 1) is primarily about exploring the (technical) potential of microservices in the organization.

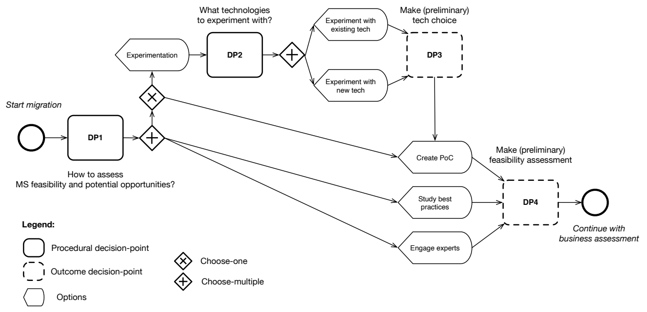

What technologies to experiment with?
Make (preliminary) tech choice
Experimentation
DP2
Experiment with existing tech
Experiment with new tech
DP3
Start migration
DP1
How to assess MS feasibility and potential opportunities?
Create PoC
Make (preliminary) feasibility assessment
Study best practices
DP4
Engage experts
Continue with business assessment
Legend:
Procedural decision-point
Outcome decision-point
Options
Choose-one
Choose-multiple

Figure 1: Technical Feasibility and Exploration of Opportunities

The first decision-point in this phase is *how to assess microservice feasibility and potential opportunities or threats* (DP1). In our interviews, we identify four options for this assessment (experimentation, building proof-of-concepts or PoCs, studying best practices, and engaging with internal or external experts). In practice, many companies will opt for a combination of multiple or all of these options.

As indicated in Figure 2, these options broadly fall into two groups: experimentation and building PoCs is highly practical and context-specific, whereas studying best practices and engaging with experts is more theoretical and conceptual. In addition, experimentation and learning from best practices are based on the knowledge that middle management and operational personnel obtain. On the other hand, launching a PoC and hiring a specialist are based on practical experiences of applications.

Experimentation is typically done early to gain understanding of the capabilities and limitations of concepts and technologies, and to assess how they apply to the specific organization, their projects, and products. If a company chooses to rely on experimentation as an outcome of DP1, they face another decision, namely *which technologies to experiment with* (DP2). Broadly, two options present themselves: (1) focusing on the technologies that the company

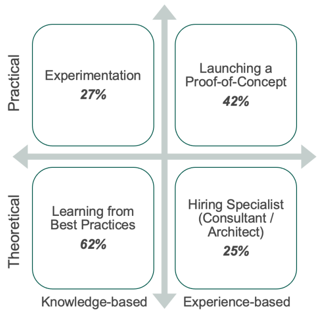


Figure 2: Options to assess microservice feasibility and potential opportunities or threats. Percentages indicate survey responses (multiple selections were possible)

already uses elsewhere and has experience in (experimenting with existing tech), (2) exploring new technologies and tools. At the end of this activity of experimentation, an outcome decision-point needs to be made, namely *which of the technology stack(s) should be selected* – at least for the time being (DP3). Worth noting is that experimentation might also be repeated later in the process, in the case of needing to evaluate more new tools or technologies. The selected technologies are then commonly the basis of one or multiple larger PoCs.

Some companies (e.g., the employers of I1 and I16) elect to skip this activity of experimentation, as the technologies to use are predecided (e.g., the company has a technology stack that it does not want to deviate from). In these cases, companies jump straight to building microservice PoCs.

Creating a PoC entails building a minimum viable version of the new system, that is then evaluated with customers or users. Additionally, it is sometimes used to demonstrate to management how the architecture will actually look like.

> "*it's kind of an investment, because initially you have to spend some money in order to do the experiments, and then you can gain the knowledge*" -I7

In addition to practical experimentation and PoC building, organizations also use more theoretical alternatives of building up microservice knowledge. One such option is the study of best practices, reports, blogs and courses that are available for self-learning. Alternatively, another way to learn is through following the documentation of frameworks dedicated for microservices (e.g. Quarkus, lagoon). Essentially, middle management and operational personnel learn by themselves about the technology. Subsequently, they act on this knowledge and transfer it to the rest of the organization.

Finally, organizations often hire specialists (or re-allocate from other parts of the company) that can transmit their knowledge and accelerate the learning process of managers and team members. For example, organizations hire experienced individuals that went through migrations many times in the past or have extensive experience on a specific framework/tool.

> "*So we have assigned architects from the different providers that we interact with and take advice from on how we could use what on the newest services*" -I9

4.1.2 Constructing the migration's business case. After technical feasibility is established, the next phase is to answer the question whether a migration also has business value, in order to engage top management and other stakeholders in middle-management that were not informed before (e.g. from marketing and sales).

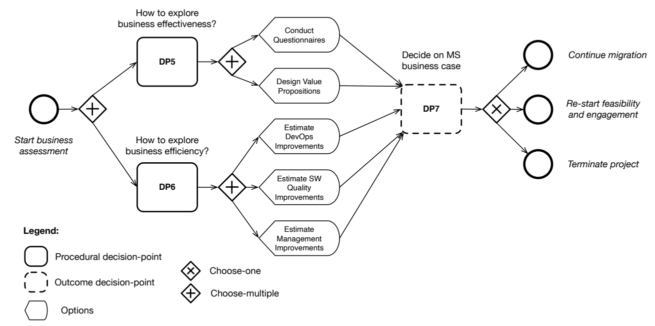

How to explore business effectiveness?
DP5
Conduct Questionnaires
Design Value Propositions
Start business assessment
How to explore business efficiency?
DP6
Estimate DevOps Improvements
Estimate SW Quality Improvements
Estimate Management Improvements
Decide on MS business case
DP7
Continue migration
Re-start feasibility and engagement
Terminate project
Legend:
Procedural decision-point
Outcome decision-point
Options
Choose-one
Choose-multiple

Figure 3: Constructing the migration's business cases

As indicated in Figure 3, organizations explore the business value of migrations along two separate axis, captured in DP5 (*how to explore business effectiveness*) and DP6 (*how to explore business efficiency*). Business effectiveness is about increasing the value that an organization delivers to its customers, typically through novel types of value-adding business services [31]. Business efficiency is about optimizing the ways in which an organization delivers value to customers and, thus, the operations in which value is created [18]. These decision-points crucially rely on the information and knowledge collected when checking on feasibility and exploring opportunities. Predominantly these decision-points serve as input for the construction and evaluating of the migration business case.

In DP5 (business effectiveness) organizations investigate how they will deliver more value to customers through their products or services. For example, organizations are able to aggregate features more flexibly and deliver a larger variety of services that are customized for their clients' needs. This can allow to sell more expensive and exclusive services, and ultimately increase profitability. Also, this allows to reach customers that before would not be reachable. We find that common ways to identify these opportunities are typically to (1) conducting customer questionnaires and (2) designing new value propositions.

Conducting questionnaires can help adopt a customer-driven approach to the migration. Our interviewees report that with questionnaires they establish communication with customers and involve them in development. This can bring a better understanding of what delivers value, and, therefore, it helps in designing new value propositions. Trying new modes of delivering value showcases to customers the value that they also gain from the migration. This enables the system's provider to sometimes even co-fund the migration with clients.

> "*we start to find the client who is more interested [...] because for sure, for them to will become faster for maintaining the database and less time for testing as well.*" -I5

Another interesting point is that microservices can enable smaller offerings that are easier to sell since this allows providing cheaper products or services with a subset of features, thus reaching smaller organizations were not targeted before.

"*It was one huge system and I could not say I'm going to deliver it in one or two days. But when I do microservices, if you are a small organization, which you need to use my system, I can do it*" -I5

DP6 (business efficiency) is about information for the business case regarding the ways in which the new microservice-based architecture will optimize the operations of the organization. Therefore, in this decision we observed options in which different bottlenecks were identified from the migration cases of this study and how they were improved. We find that there are three broad classes of possible improvements, some or all of which can be explored in this phase: improvements in operations and DevOps (e.g., more efficiency in systems operation), improvements in software quality (e.g., better performance at scale), and improvements in management processes (e.g., through easier recruitment of highly skilled individuals).

The information from all options that the organization decided to explore in DP5 and DP6 feed into the outcome decision-point DP7 (*deciding on a the microservices business case*). Here, one or multiple business cases are constructed based on all information collected so far. The business case(s) facilitate the deliberation between stakeholders, and are used for comprehending and aligning on both the business and technical logic of the migration. Three alternatives present themselves as outcome of this business case construction and assessment: either to continue with the migration as planned (leading to commencing the technical migration), termination of the project (if obstacles are identified that seem not possible to overcome), or to revisit the decisions of technical feasibility and exploration of opportunities (e.g., if there appears to be potential, but additional information needs to be collected, for instance by assessing new technology or exploring additional value propositions).

"*the best way is to say in the very beginning: Okay, this is how much it cost us now, [...] and then you say: Okay, if we move this out, then we would need that many resources [...] and costs should go down [...] and you can also scale up and down*" -I10

The business case demonstrates from a business point of view how a microservice-based architecture can increase the focus on value adding activities, instead of repetitive tasks and short-term fixes. The interviews indicated to different aspects for improvement potentials on Business Effectiveness and Business Efficiency. We validate these aspects with the participants of our survey and we present the most frequent ones in Table 2 and Table 3.

Measure of Effectiveness	*Selected in Survey*
Quality & experience improvements	65%
Aggregating services to new offerings	37%
Focused / specialized offerings	26%

Table 2: Improvements in terms of business effectiveness selected by more than 25% of the survey respondents. Multiple selections were possible.

4.2 Decisions on the Technical Dimension

Once the business case for migrating towards microservices is made, the organization engages the technical migration. This requires a number of additional decisions to be taken, which are summarized in Figure 4. The decisions on the technical dimension are validated

Measure of Efficiency	*Selected in Survey*
Scalability	67%
Maintainability	54%
Continuous deployment	44%
Flexibility in skill	37%
Lightweight application	33%
Less duplicated functionality	33%
Robustness & rigidity of system	29%
Cost reduction of development	25%

Table 3: Improvements in terms of business efficiency selected by more than 25% of the survey respondents. Multiple selections were possible.

through the survey and the percentage depicted in the figure indicate how often the different options have been chosen by the survey respondents.

The first technical decision-point is DP8 (*what is the high-level migration strategy*). Specifically, the migration cases we investigated revealed three alternative approaches to migrating a system towards microservices.

One alternative is to start the migration with architecting and developing a new system on the green field (from scratch). This means that the monolith is not altered in any way and a new system is developed entirely from the beginning. This case includes radical refactoring activities and usually leads to simultaneously developing and operating two systems until the microservice system is ready and the monolith can be turned off. Interestingly, this is the least popular option across survey respondents, with only 12% choosing it.

On the other hand, the options are gradually decomposing the monolith, retaining the monolith and building all new features as microservices or a combination of decomposing and building new features. These option result in a hybrid architecture, in which a small monolith (a "microlith") lives within the microservice-based system. The first option requires the extraction of microservices until the monolith is decomposed. This approach is about repeatedly extracting from existing functionality pieces of code that can be individual microservices, until the entire system is decomposed. In addition, this approach is the most popular across survey respondents with 45% choosing it. The second option is developing every new feature as a microservice and then operating a hybrid architecture until the old monolith phases out (chosen by 14% of survey respondents).

Finally, we find that a common choice (confirmed by 29% of survey respondents) is to combine gradual extraction with the retention of a "microlithic" kernel of old features. Those are for example features that are difficult or impossible to extract as microservies. In this case, all new features are developed with microservices principles in mind and at the same time some features are extracted gradually, until the application reaches a level of granularity that is considered sufficient. More often than not, interviewees mention that the perfect microservices-based architecture cannot be achieved and the required mindset is to accept this and constantly improve the system.

"*I don't think it's black and white where before we had a monolith and now we have a complete microservice architecture. It's more like a gradual process, that might never be completed*" -I5

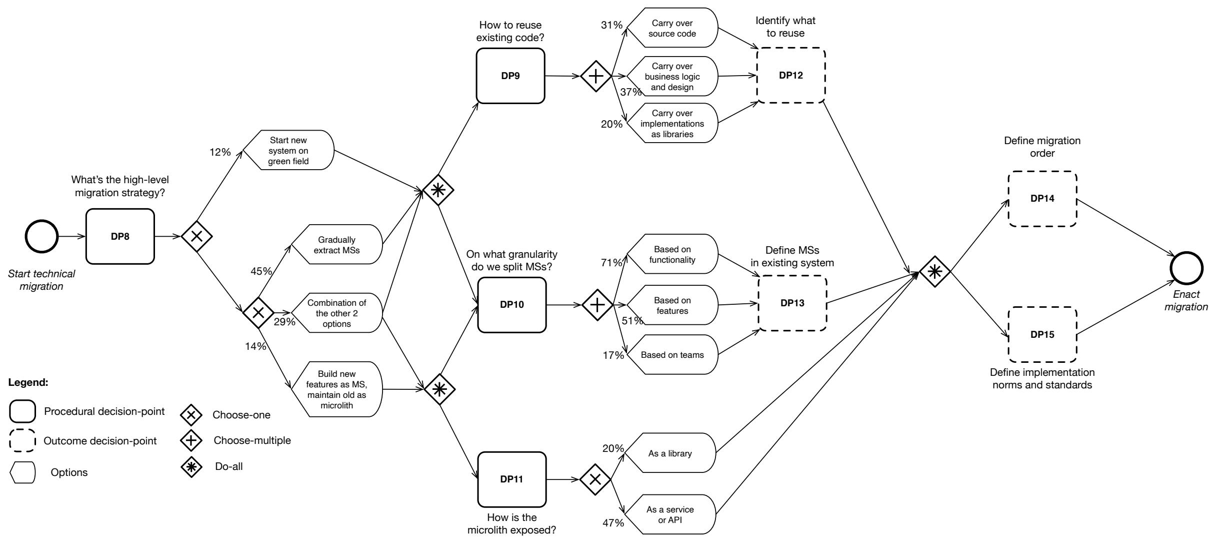

Start technical migration
What's the high-level migration strategy?
DP8
12%
Start new system on green field
45%
Gradually extract MSs
29%
Combination of the other 2 options
14%
Build new features as MS, maintain old as microlith
How to reuse existing code?
DP9
31%
Carry over source code
37%
Carry over business logic and design
20%
Carry over implementations as libraries
Identify what to reuse
DP12
On what granularity do we split MSs?
DP10
71%
Based on functionality
51%
Based on features
17%
Based on teams
Define MSs in existing system
DP13
DP11
How is the microlith exposed?
20%
As a library
47%
As a service or API
Define migration order
DP14
DP15
Define implementation norms and standards
Enact migration
Legend:
Procedural decision-point
Outcome decision-point
Options
Choose-one
Choose-multiple
Do-all

Figure 4: Technical decisions for migrating towards microservices

Once the high-level migration strategy is decided, organizations now reach decision-point DP9, and need to assess *how to reuse existing code* (if anything shall be re-used at all, and if the old system is not simply re-used in it's entirety as a "microlith" using the third option from above). Three different options for re-use have emerged in our study (which can evidently be combined). Re-using source code (e.g., by copying-and-pasting useful code), carrying over business logic and software designs that business analysts developed from requirements, or by encapsulating key functionality in libraries, which are then imported in the new microservices. Carrying over business logic and designs is the most common option, with 37% of respondents doing so. Carrying over implementations as libraries is the least common option with 20% of respondents doing so. Source-code is carried over by 31% of respondents. It is worth noting that even organizations that started from scratch often reused some artifacts in that manner, though usually not source code. This then leads to outcome decision-point DP12, which requires the organization to *identify what concretely shall be reused.*

Organizations that decide to develop all new functionality as microservices and leave the existing system as a "microlith" within the new architecture face a different decision-point (DP11), and need to decide *how to expose the "microlith"*. The two observed choices to this end entail exposing the "microlith" as a library (similar to the re-use of other, smaller, code elements) or to host the "microlith" as a service or API. It is worth noting that all survey respondents that had a "microlith" to expose they made use of a shell API.

> "*First step was to take the back-end as a whole, as one piece out of the front-end and connect them with one big library that is imported in the UI. And then we built on API around it and that's where we could have a back-end and the front end.*" -I11

Independently of their choice in DP8, an organization now also needs to decide *on which granularity microservices shall be split* (DP10). For organizations starting on a green field or which gradually extract microservices, this decision applies to the services representing existing code. However, even if the old system shall be retained as a "microlith" this decision needs to be made for new services. The first option here (chosen by 71% of survey respondents) is to split services based on the functionality of the microservice, i.e., based on functions (for example sorting) or architectural layers (for example front-end). The second option (chosen by 51% of survey respondents) is to split microservices based on features, i.e., designing microservices that are small end-to-end applications that go through the entire development stack. The third option (chosen by 17% of survey respondents) is to align the service structure with existing team structures. For example, if teams are in different geographic locations and time-zones and work independently on different tasks, the microservices are designed to accommodate this particularity. Finally, a rule of thumb we observed is achieving the required level of granularity by keeping a microservice independent as long as the code required for communication is smaller than the code required to implement the microservice.

> "*There's always an amount of code and logic needed, just to get a service up and running [...] and if it is bigger than the actual business logic part of the service, then [...] it is too small and I figure out where it can fit instead.*" -I15

At this point in the migration the organization has made most planning-related decisions. What is still left to do are three outcome decision-points, where the organization takes in all decisions and lessons learned so far to decide on the actual structure of their new system. These decisions start with DP13 (*define what microservices the new system should contain*).

Then in DP14, (*define the order of migration*) we accumulated the most common activities of migrations and we order them chronologically, as shown in Figure 5. We observe from the interviews 6 main activities that constitute a migration and we ask survey respondents to sort them in chronological order. This order showcases a tendency of engineers to order migrations in splitting the back-end, then the database, then DevOps, then microservices communications and finally reorganizing teams.

Finally, DP15 (*define new implementation norms and standards*) allows to establish the governance for scaling the migration to cover most parts of the system.

> "*For it to make sense, you want it to scale across the organization and across the enterprise. And then you need to have kind of a strong governance within that organization*" -I13

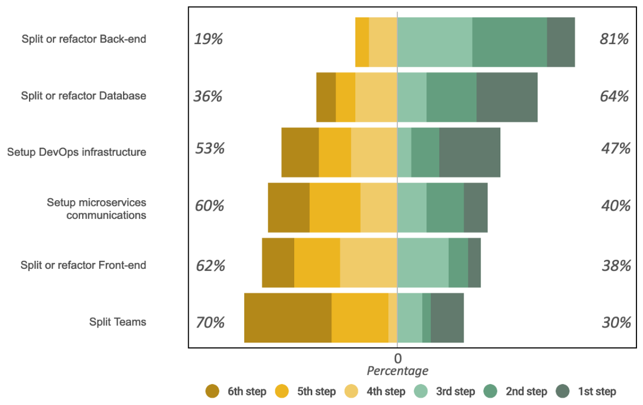


Figure 5: Distribution of ordered migration activities (by survey respondents) showing the order of migration based on popularity

Furthermore, these final decisions entail also concerns that interviewees thought are important to address early in the migration, since many migration cases showed evidence that neglecting or postponing some aspects entail risks.

> "*There were some things that should be said from the very beginning in order to avoid latter additional efforts.*" -I2

For example, I7 mentions security and how it became a challenge to address later on if neglected in the migration.

> "*security is neglected from the start, making it hard to add it later on*" -I17

Another example is how not re-developing logging mechanisms and exception or error handling leads to costly future work.

> "*We started in the beginning, doing the development without deciding what the type of our logs will be, how we'll do the exception handling [...] and now someone should go back to all the development that we did and put logs and put the correct errors and exceptions.*" -I2

4.3 Decisions on the Organizational Dimension

Moreover, a migration entails a set of decisions that are taken to restructure the organization as well as the organization's operations. The objective is typically to lead operations in the direction of achieving the estimated benefits from the business case. Another objective is to align business with the new capabilities and requirements of the new technology, that come from the decisions of Section 4.2. We identify the constituent organizational decision-points, as summarized in Figure 6.

The first organizational decision-point is DP16 (*how to drive the migration and maintain control on it*). A migration typically takes a significant amount of time to complete and organizations use two options to maintain a stable track towards the new architecture. One option observed is to distribute responsibilities to developers. The rationale is usually to distribute the migration activities in the entire organization. Hence, organizations avoid the risk of having isolated knowledge for the migration, that is not systematized or validated throughout the entire organization. The other option is to dedicate a team specifically responsible for the migration. This can eliminate the perception that the migration is a project on the side, since a specialized team makes the project a main contributor to the organization's strategy. The rationale is also to accelerate the learning curve of the organization on MSA, by dedicating a specialized team to the new technologies.

> "*Some team needs to work on the the new product and you need to allocate some people and to invest with them*" -I4

These two options also seem to relate on the mutual understanding between stakeholders on how to make the migration's decisions. This is basically the decision-making process of the migration towards microservices, which normally requires consensus building and alignment between varied stakeholders. This consensus can either be established top-down (through management decisions, e.g., in case of I12) or bottom-up (through extensive team discussion and deliberation, e.g., I2 or I3). Once it is clear how the migration will take place, organizations reach decision-point DP17 (*how the organization is getting restructured*), which has three options.

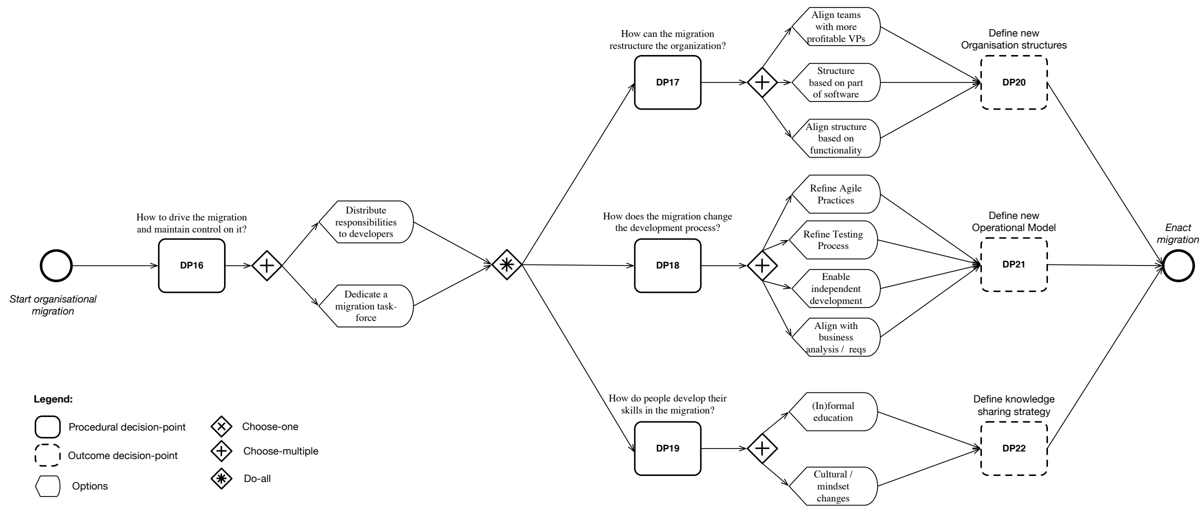
The first option on restructuring the organization is by aligning the teams' structures with the most profitable value proposition. For example, I5 mentions that if at a given point of time there is a need for engineers in a very profitable project, then people from different teams are assigned to it. I5 also mentions that this dynamism happens quite regularly and that decisions are made with an eye on immediate customer value.

> "*We started to present and see what is easier for them, especially from customer services, [...] if there's something that they [the customers] are more interested in.*" -I5

The second option is to align team-structures based on parts of the software developed. For example, I11 distinguishes front-end microservices from back-end microservices, having teams assigned to each part. Finally, there is the third option with a way of organizing roles, responsibilities and ownership of services based on the functionality and the features developed.

> "*break the team into smaller parts and say you and you will support this service, you and you will support the other services"*" -I3

Also, organizations reach the decision-point DP18 (*how does the development process change*). Organizations optimize operations by simplifying their process, for example through enabling parallel development in parts of the code-base or with faster identification of issues. We identify four options of altering the development process. One way is through refining the agile practices adopted from the organization. Another option is through refining the testing process. Additionally, another option way of process change is enabling the independent development from developers. Finally, an identified option is aligning with business analysis and ways of extracting or communicating requirements throughout the process.



Start organisational migration
How to drive the migration and maintain control on it?
DP16
Distribute responsibilities to developers
Dedicate a migration task-force
How can the migration restructure the organization?
DP17
Align teams with more profitable VPs
Structure based on part of software
Align structure based on functionality
Define new Organisation structures
DP20
How does the migration change the development process?
DP18
Refine Agile Practices
Refine Testing Process
Enable independent development
Align with business analysis / reqs
Define new Operational Model
DP21
How do people develop their skills in the migration?
DP19
(In)formal education
Cultural / mindset changes
Define knowledge sharing strategy
DP22
Enact migration
Legend:
Procedural decision-point
Outcome decision-point
Options
Choose-one
Choose-multiple
Do-all

Figure 6: Restructuring the organizational structures and operations

"*the guys that were writing the functional specs, you know the business requirements basically, they have to somehow map their ideas and their designs to specific microservices somehow*" -I3

We evaluate through the survey how the options from DP17 and DP18 resonate with developers and present a summary of responses in Figure 7.

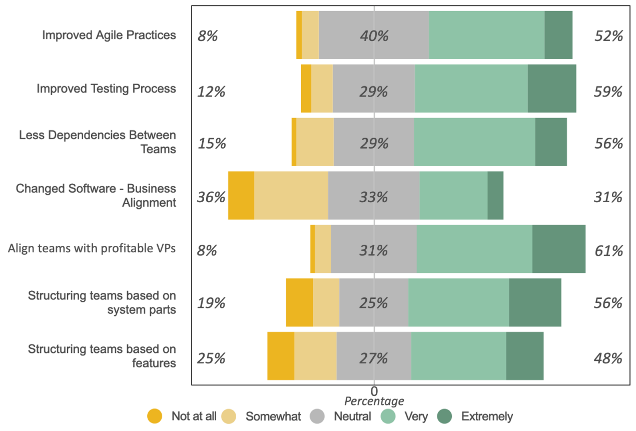


Figure 7: Diverging bar chart on the options in which operations change and organizations restructure (ordered as in Figure 6)

The next decision-point is DP19, or *how do people develop their skills and knowledge in migrations?* Many interviewees mentioned cultural changes that need to take place and the learning curve required to take in order to succeed in the migration.

"*People needed to educate and train a lot. So, the company invested on that one as well, to train people*" -I3

We identify two options for DP19, coming from the fact that migrations change the expectations on engineers and management in terms of knowledge. The first option is to provide (in)formal education of the new technologies, tools and frameworks. Most importantly, the second option is giving time to experience the new architecture and develop knowledge in terms of learning a new paradigm of development and transforming the mindset and culture accordingly.

"*Also, maybe change of the mindset of some people as well. Both design people and developers*" -I3

Finally, the decision-making process of the organizational and operational restructuring concludes with three outcome decision-points. In DP20 organizations *define the new organizational structures* and in DP21 *define the new operational model.* DP22 is to *define the knowledge sharing strategy of the organization* and how skills will be propagated across the organization.

5 DISCUSSION

Decision-making typically happens in an individual level, group level and organizational level. The focus of our findings is on the complex decisions that take place on an organizational level.

Multidimensionality of migrations: First, our findings show that migrations towards microservices are not only about technological change, but also about other dimensions. For example, microservices often facilitate potential cross-team collaboration and autonomy of teams. This, in combination with how teams structures change, shows us how microservices are not only a software architecture but also an enabler of organizational restructuring. Migrations are long procedures that entail changes in business, organization, culture, mindset, roles, skills and of course, technology. The dimensions we identify are in line with socio-technical systems engineering, containing organizational structures (Section 4.3), processes (contents of decision-making processes), technologies (Section 4.2) and of course people (stakeholders) [7]. Additionally, we demonstrate how to achieve comprehensive strategic alignment to leverage microservices for delivering strategic value

[18]. Specifically, this takes place by including the aspect of business effectiveness [31] in decision-making and considering how it will increase the delivery of value to customers and users. For example, through new sales models and also through the delivery of new business value from aggregating different combinations of services. The aspect of business effectiveness complements existing work on the economics of microservices that demonstrate how the architecture improves efficiency [27].

Human-centered approach in microservices migrations: Adopting a value-based approach can help to accommodate the needs of all stakeholders and achieve stronger engagement. Decision-making mechanisms of migrations are often not centralized, spanning from board-room level, to operational level. Some of the most complex and under-studied aspects during migrations are the dynamics between stakeholders in decision-making. We address this gap by aggregating a set of decisions that organizations make during migrations. Our theory contributes by providing a process for understanding the different perspectives and incentives that usually prevail among different stakeholders. We evolve existing knowledge of decision-making in different stakeholders views [21], by demonstrating in detail these decisions and the point in which they are made.

For example, to ensure the engagement of key stakeholders from within the organization it is essential to consider both their business-oriented or technical-oriented backgrounds and roles. A business case is the framework of demonstrating the value that a microservices-based architecture can bring to different stakeholders. The demonstrated value ranges from aspects like increasing profitability of products and services, to potentially tackling talent scarcity (due to polyglot nature) and fast on-boarding of new employees (microservices are, at least in theory, small and easily comprehensible).

A pragmatic view on microservices migrations: In addition, a common view of all interviews was that microservices is not a silver bullet and their benefits need to be justified properly and as early as possible. Otherwise, organizations might unjustly choose to migrate in cases that do not fit with the architecture. This can lead to failed migrations and a negative attitude towards the technology, despite the fact that in different cases from those attempted the architecture might be a better fit. In addition, some organizations consider the technical overhead that comes with microservices not worthy of the effort in certain cases. Even though a MSA has many benefits, organizations should still consider alternatives like keeping a monolith, purchasing a software-as-a-service, or outsourcing development.

Moreover, our decision-making processes are modeled as linear, mainly for visualization purposes of showing the points in which decisions need to be made. However, these decision-making processes have a more iterative nature in practice, i.e., they re-occur during a migration. It is important to maintain a dynamism in teams during migrations. Many interviewees even mentioned the need to designing the system with flexibility, giving space to update, extend and modify it. The core of migration projects is continuously unravelling or highlighting architectural design inefficiencies and actively eliminating them, through decision-making. For example, when organizations decide what to reuse, they make a selection of designs that are worthy to be in the system, throwing away the rest.

Relations and dependencies among decisions: We also identify the typical activities that a migration entails and a common order of these activities. The different ways in which organization evaluate and explore the value from microservices can influence the course that the migration will eventually take. For example, an organization that starts by exploring the potential of microservices with experimentation and a PoC leads the migration to a particular path. This can help to gain more accurate estimates of tasks or of the technical capabilities and limitations. Hence, planning can happen with a better understanding of the time needed and the potential risks. In addition, early decisions can sometimes influence also subsequent decisions. For example, organizations that validate novel value with customers, can manage to strongly engage top management, increasing the support to the migration.

Also, the three predominant options of the technical dimension that we observe is similarly to existing industry practices [25]. The approach of re-developing a system on a green-field seems quite radical but has some advantages of having a clean architecture more easily. On the other hand, gradually decomposing a system enables better usage of older artifacts and therefore, faster delivery. Developing only new features as microservices can also take place in software that has short life span in usage and the monolith will naturally be outdated at some point in time. These approaches also hint on the differences between "gradually dialing" to microservices versus "radically jumping" to microservices.

6 CONCLUSION

In this paper, we investigate how software engineers and organizations go through migrations towards microservices. Specifically, we obtain an understanding on the different decisions involved in migrations along with their interplay. The 16 different cases of migrations towards microservices that we investigate in this study reveal to us details on the different dimensions of decision-making during migrations. On one hand, a set of decisions take place in order to prove the technical feasibility of a microservices-based architecture. On the other hand, there are decisions regarding the driving forces for pursuing a migration on the first place and how engagement is achieved. This shows how to create the required "buy-in" in order to make a microservices migration get in grips across the organization. In addition, we see decision-points that take place on the technical dimension and how these resonate with decision-points that take place in an organizational and operational level. Importantly, we see how these dimensions are complementing each other to motivate migrations more strongly and make more aware steps of change. The decisions taken during the migration towards MSA along with their alternative options are partially validated via a survey that 52 professionals responded to.

ACKNOWLEDGMENTS

This work was done as part of TrAF-Cloud funded by the Swedish innovation agency VINNOVA (proj.no. 2018-05010). We would also like to express our appreciation to all the participants of the study for their time, support and valuable insights during the interviews and the survey.

REFERENCES

[1] 2021. *Decision-Making in Microservices Migrations.* Zenodo. https://doi.org/10.5281/zenodo.4561793

[2] Deepika Badampudi, Krzysztof Wnuk, Claes Wohlin, Ulrik Franke, Darja Smite, and Antonio Cicchetti. 2018. A decision-making process-line for selection of software asset origins and components. *Journal of Systems and Software* 135 (2018), 88–104. https://doi.org/10.1016/j.jss.2017.09.033

[3] Armin Balalaie, Abbas Heydarnoori, and Pooyan Jamshidi. [n.d.]. Microservices architecture enables devops: an experience report on migration to a cloud-native architecture. *IEEE Software* 33 ([n. d.]), 1–1.

[4] Armin Balalaie, Abbas Heydarnoori, and Pooyan Jamshidi. 2016. Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture. *IEEE Software* 33, 3 (2016), 42–52. https://doi.org/10.1109/MS.2016.64

[5] Armin Balalaie, Abbas Heydarnoori, Pooyan Jamshidi, Damian A. Tamburri, and Theo Lynn. 2018. Microservices migration patterns. *Software - Practice and Experience* 48, 11 (nov 2018), 2019–2042. https://doi.org/10.1002/spe.2608

[6] Sebastian Baltes and Paul Ralph. 2020. Sampling in Software Engineering Research: A Critical Review and Guidelines. *CoRR* abs/2002.07764 (2020). arXiv:2002.07764 https://arxiv.org/abs/2002.07764

[7] G. Baxter and I. Sommerville. 2011. Socio-technical systems: From design methods to systems engineering. *Interacting with Computers* 23, 1 (2011), 4–17. https://doi.org/10.1016/j.intcom.2010.07.003

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

[9] Kathy Charmaz. 2014. *Constructing grounded theory.* sage.

[10] P. Di Francesco, P. Lago, and I. Malavolta. 2018. Migrating Towards Microservice Architectures: An Industrial Survey. In *2018 IEEE International Conference on Software Architecture (ICSA).* 29–2909. https://doi.org/10.1109/ICSA.2018.00012

[11] Paolo Di Francesco, Patricia Lago, and Ivano Malavolta. 2019. Architecting with microservices: A systematic mapping study. *Journal of Systems and Software* 150 (2019), 77–97. https://doi.org/10.1016/j.jss.2019.01.001

[12] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. 2017. *Microservices: Yesterday, Today, and Tomorrow.* Springer International Publishing, Cham, 195–216. https://doi.org/10.1007/978-3-319-67425-4_12

[13] Nicola Dragoni, Ivan Lanese, Stephan Thordal Larsen, Manuel Mazzara, Ruslan Mustafin, and Larisa Safina. 2018. Microservices: How To Make Your Application Scale. In *Perspectives of System Informatics*, Alexander K. Petrenko and Andrei Voronkov (Eds.). Springer International Publishing, Cham, 95–104.

[14] J. Fritzsch, J. Bogner, S. Wagner, and A. Zimmermann. 2019. Microservices Migration in Industry: Intentions, Strategies, and Challenges. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME).* 481–490. https://doi.org/10.1109/ICSME.2019.00081

[15] Jonas Fritzsch, Justus Bogner, Alfred Zimmermann, and Stefan Wagner. 2018. 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, 128–141.

[16] Michael Gysel, Lukas Kölbener, Wolfgang Giersche, and Olaf Zimmermann. 2016. Service Cutter: A Systematic Approach to Service Decomposition. In *Service-Oriented and Cloud Computing*, Marco Aiello, Einar Broch Johnsen, Schahram Dustdar, and Ilche Georgievski (Eds.). Springer International Publishing, Cham, 185–200.

[17] Sara Hassan, Rami Bahsoon, and Rick Kazman. 2020. Microservice transition and its granularity problem: A systematic mapping study. *Software - Practice and Experience* 50, 9 (2020), 1651–1681. https://doi.org/10.1002/spe.2869 arXiv:1903.11665

[18] J. C. Henderson and H. Venkatraman. 1999. Strategic alignment: Leveraging information technology for transforming organizations. *IBM Systems Journal* 38, 2.3 (1999), 472–484. https://doi.org/10.1147/SJ.1999.5387096

[19] John P Kotter et al. 1995. Leading change: Why transformation efforts fail. (1995).

[20] John P Kotter and Leonard A Schlesinger. 1979. Choosing strategies for change. (1979).

[21] Philippe Kruchten, Rafael Capilla, and Juan Carlos Duenas. 2009. The Decision View's Role in Software Architecture Practice. *IEEE Software* 26, 2 (mar 2009), 36–42. https://doi.org/10.1109/MS.2009.52

[22] Jyhjong Lin, Lendy Chaoyu Lin, and Shiche Huang. 2016. Migrating web applications to clouds with microservice architectures. In *2016 International Conference on Applied System Innovation, IEEE ICASI 2016.* Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICASI.2016.7539733

[23] Genc Mazlami, Jurgen Cito, and Philipp Leitner. 2017. 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., 524–531. https://doi.org/10.1109/ICWS.2017.61

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

[25] Sam Newman. 2019. *Monolith to microservices: evolutionary patterns to transform your monolith.* O'Reilly Media.

[26] Zhongshan Ren, Wei Wang, Guoquan Wu, Chushu Gao, Wei Chen, Jun Wei, and Tao Huang. 2018. Migrating Web Applications from Monolithic Structure to Microservices Architecture. In *Proceedings of the Tenth Asia-Pacific Symposium on Internetware* (Beijing, China) *(Internetware '18).* Association for Computing Machinery, New York, NY, USA, Article 7, 10 pages. https://doi.org/10.1145/3275219.3275230

[27] A. Singleton. 2016. The Economics of Microservices. *IEEE Cloud Computing* 3, 5 (2016), 16–20. https://doi.org/10.1109/MCC.2016.109

[28] Klaas Jan Stol, Paul Ralph, and Brian Fitzgerald. 2016. Grounded theory in software engineering research: A critical review and guidelines. *Proceedings - International Conference on Software Engineering* 14-22-May-2016, Aug 2015 (2016), 120–131. https://doi.org/10.1145/2884781.2884833

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

[30] Johannes Thönes. 2015. Microservices. *IEEE software* 32, 1 (2015), 116–116.

[31] N. Venkatraman and Vasudevan Ramanujam. 1986. Measurement of Business Performance in Strategy Research: A Comparison of Approaches. *Academy of Management Review* 11, 4 (1986), 801–814. https://doi.org/10.5465/amr.1986.4283976 arXiv:https://doi.org/10.5465/amr.1986.4283976

[32] Uwe Zdun, Erik Wittern, and Philipp Leitner. 2020. Emerging Trends, Challenges, and Experiences in DevOps and Microservice APIs. *IEEE Software* 37, 1 (jan 2020), 87–91. https://doi.org/10.1109/MS.2019.2947982

[33] Olaf Zimmermann. 2017. Microservices tenets: Agile approach to service development and deployment. *Computer Science - Research and Development* 32, 3-4 (jul 2017), 301–310. https://doi.org/10.1007/s00450-016-0337-0