How Does Migration to Microservices Happen? A Survey of the Finnish Software Industry

Kristian Tuusjärvi[1,*,†], Jussi Kasurinen[2,†] and Sami Hyrynsalmi[3,†]

[1]*Lappeenranta University of Technology, Yliopistonkatu 34, 53850 Lappeenranta, Finland*
[2]*Lappeenranta University of Technology, Yliopistonkatu 34, 53850 Lappeenranta, Finland*
[3]*Lappeenranta University of Technology, Yliopistonkatu 34, 53850 Lappeenranta, Finland*

Abstract
When a software system gets old, it is time to decide whether it is replaced, revised, or removed from the support life cycle. In modern software development, migrating old monolithic systems into a solution by applying microservices is an option when the system is still needed, or the decision is made to modernize the existing component with new technology. In this study, we conducted an online survey on 28 Finnish software professionals who provide MSA migration services to their clients. We aimed to understand how these MSA migration projects happen, how they are resourced, what skills are useful for this type of work, and how common these activities are. Based on our observations, migration projects usually replace monolithic systems with less than ten services and require advanced software engineering skills and an understanding of trade-offs in software architectures. With these results, we can continue our work towards qualitative studies to compare general observations against more practical issues of MSA migration processes.

Keywords
Microservice architecture, Modernization, Migration, Online survey,

1. Introduction

In this research paper, we describe our survey study on the migration of legacy systems. Specifically, we focus on migrating monolithic legacy software to microservice architecture (MSA). MSA has gained traction in the past two decades due to technological advancements such as cloud computing and containerization, offering scalability, cost-effectiveness, and faster development cycles. The increasing digitization of businesses drives the search for more dynamic and adaptive software architectures.

Legacy systems refer to old systems that have been neglected, lack documentation, tests, and proper maintenance, and often have poor architecture. Despite their deficiencies, these systems are maintained because they are necessary for a business to continue operating. In essence, while legacy systems may not be ideal from a software development perspective, they are crucial for the ongoing operation of a business and, therefore, must be kept running [1].

Unlike a traditional monolithic system, MSA is a software architecture style that uses small decentralized services that interact with each other using messaging protocols, such as Representational State Transfer (REST) [2, 3, 4]. In the early 2010s, MSA started to rise in popularity [5]. It was popularized by large software companies such as Sound Cloud [6], Netflix [7], and Uber [8]. MSA's main quality attributes are availability, flexibility, maintainability, scalability, and loose coupling, [9, 10] which are desirable for large software systems developed by the aforementioned companies.

Large legacy systems are often migrated to MSA because systems developed with MSA are less likely to accrue complexity throughout their lifespans. MSA systems are also more decoupled, meaning that parts of the system can be developed more isolated from the rest of the system, which decreases the dependencies between the software components. This allows the software components to be maintained apart, letting the whole system stay robust and responsive [4].

Our previous SMS study [11] into this subject found many challenges associated with the re-engineering process: lack of decomposition approaches, high level of coupling, lack of guidelines, identification, and boundary recognition of microservices. Related to the motivations, we found scalability, maintainability, time to market, and adaptability to new technologies. Based on these findings and the related research, we wanted real-world results from the Finnish software industry, as much of the research is conducted abroad in different business areas. The rest of this research paper is structured: study design, related research, results, discussion, threats to validity, and conclusion.

TKTP 2024: Annual Doctoral Symposium of Computer Science, 10.-11.6.2024 Vaasa, Finland
*Corresponding author.
†These authors contributed equally.
kristian.tuusjarvi@gmail.com (K. Tuusjärvi); jussi.kasurinen@lut.fi (J. Kasurinen); sami.hyrynsalmi@lut.fi (S. Hyrynsalmi)
https://ktcoding.fi (K. Tuusjärvi); https://www.jussikasurinen.net/about.html (J. Kasurinen)
0009-0008-3974-4038 (K. Tuusjärvi); 0000-0001-9454-8664 (J. Kasurinen); 0000-0002-5073-3750 (S. Hyrynsalmi)

2. Study design

This study aims to understand the trend of modernizing legacy software to MSA and to add to the empirical research studying migrations from legacy systems toward MSA. Specifically, we want to research practitioners' core challenges in the Finnish software industry to understand their environment, what motivates modernization to MSA from the legacy system, and the challenges of different phases of modernization. Additionally, we wanted to gather information about the organizations and projects that attempt MSA migration. The research questions below reflect our research goal.

- RQ1: What are the project details of MSA projects?
- RQ2: What are the success factors and methodologies for MSA migration?
- RQ3: What are the main challenges and motivations in migrating to MSA?

To better map the challenges of the migration process, we used a horseshoe model of software modernization. The horseshoe model includes three phases. First is the reserve engineering of the existing system, meaning the mapping and understanding of the existing system to plan for the transformation of the system. The second phase is transformation, which means altering the existing system to the desired state. Finally, the third phase is forward engineering, meaning making the changes described in the second phase. [12] Regarding the research design process, we decided on the research questions we wanted to use to base the survey. After that, we started an iterative process to generate the survey questions. The iterative process of generating the survey questions was done mainly through email exchanges, going back and forth until we were content with the content of the preliminary survey. To test the survey, we sent it to a few experts in the field to review and gather feedback. After fixing the suggestions from our experts, we launched the survey online. The survey consists of 20 questions, with only one open-ended question. The rest have options, and we also included an 'other' option if none fit. We grouped the questions into project details, success factors, challenges, and motivations-related questions. Project details questions reflected what kind of people and teams were working on the MSA project. Success factors questions queried their choices in different situations. Finally, the challenge and motivation-specific questions focused on the MSA project's challenges and motivations. The survey was published online and was available through a web link. Our survey's target audience was practitioners with at least some experience working with legacy to MSA modernization. We recruited respondents through social media such as Linkedin and X (formerly Twitter). The goal was to find a person with information about MSA on their social media page and then approach them about filling out the survey. We also used our connections to reach out to people with expertise related to the topic. To increase the effect, we also asked our connections to ask about their connections in a snowballing fashion. We also attempted to reach out to people on social media who had advertised their expertise in this field; this approach was efficient. We analyzed the data using spreadsheets and charts to interpret the results. We received 28 responses to the survey. We have published the results and the survey template to Figshare (https://dx.doi.org/10.6084/m9.figshare.25487269) for repeatability purposes.

3. Related research

This section will discuss the related research focusing on the industry challenges and motivations for migrating toward MSA.

Taibi et al. [13] conducted a survey study in 2017 where they interviewed 21 practitioners who had migrated to MSA in the past two years. Similar to our research, they wanted to know the motivations, advantages, and disadvantages of migrating to MSA. The primary challenges that they identified were monolith decoupling, database migration, and communication between microservices. The primary benefits of MSA are maintainability, scalability, return on investment (ROI), and complexity reduction. They conclude that scalability and maintainability are primary drivers for migrating to MSA. Their study method differs from ours as we deployed an online survey, whereas they used in-person interviews. We also gather other data related to general system information and organizational and demographic data.

In their study, Ghofrani et al. [14] surveyed 25 experts to understand the challenges and practices associated with microservices architecture (MSA). The survey identified several challenges, including the distributed nature of MSA, skill and knowledge gaps, and domain separation and service boundary definition difficulties. The primary reasons for adopting MSA were scalability and agility, with manual methods commonly used to determine service boundaries. The surveyed experts also expressed concerns about security, licensing, and memory usage and suggested improved notations, techniques, and frameworks for MSA design. They also identified security optimization, response time, and performance enhancement as the top priorities for improvement. While Ghofrani et al. [14] also study the challenges of microservices, they do not specifically focus on the migration process as we do. Furthermore, they researched the barriers to MSA usage and the solutions to improve MSA, which we have not researched.

Francesco et al. [15] conducted a survey in 2018 on the challenges of migrating applications to microservices architecture. Experienced IT practitioners were surveyed and interviewed. Challenges included releasing new features, high coupling in legacy systems, and difficulties in testing and maintenance. The research by Francesco et al. [15] is similar to ours as it is a survey study that also studies the challenges of migrating to MSA and utilizes the horseshoe model. However, they have not researched the motivation for migrating.

Fritzsch et al. [16] conducted an interview study in 2018 to research MSA migration strategies, motivations, and challenges. The research explores the migration of 14 systems to microservices based on 16 in-depth interviews with professionals from 10 companies in Germany. The key drivers for migration were maintainability and scalability. Challenges faced included service decomposition difficulty, a shortage of Microservices expertise, and organizational hurdles [16]. Their research differs from ours in that interviews are used, not surveys. Additionally, they focus on migration strategies, which we do not do.

In their study (2018), Bogner et al. [17] interviewed 17 software professionals from 10 companies in Germany to evaluate 14 service-based systems' adherence to Microservices characteristics and their impact on software quality. The study revealed that migration to Microservices was primarily driven by the need to improve maintainability and that HTTP and Docker containers were preferred technologies. At the same time, Microservices positively or neutrally impacted software quality [17]. Compared to our research, they focus more on the MSA technologies, qualities, and quality. Furthermore, our research is based on a survey study while they conducted an interview study.

In 2018, Knoche et al. [18] surveyed 71 German professionals to determine the primary drivers and barriers to adopting MSA, the goals of modernizing MSA, and how data consistency affects performance. They concluded that scalability, maintainability, and time to market were the main drivers for modernization, with developer's and staff's skills being the main barriers. Early adopters desired scalability, while traditional companies wanted maintainability [18]. Their study is similar to ours as they also research the motivations and goals of MSA migration. However, unlike us, they researched the barriers preventing MSA adoption and the runtime effects of MSA.

Some research has been conducted to study the motivations and challenges related to MSA migration [14] [15] [16] [18]. However, there are some differences, namely the study type used. Additionally, the research is primarily regional, focusing on Germany [14] [16] [17] [18]. Our research brings a new region into the current research field to complement the existing research findings.

4. Results

4.1. Project details

In this section, we describe the demographics of the survey respondents. The respondents have a significant amount of experience in software development. Most respondents (60.7 %) have more than ten years of experience, and 89 % of respondents have four years or more experience in software development, Table 1.

Table 1
How many years of experience you have on software development, or related areas?

Choice	#	%
Less than 1 year	0	0.0%
1-3 years	3	10.7%
4-6 years	5	17.9%
7-10 years	3	10.7%
More than 10 years	17	60.7%

We can observe the relative novelty of MSA-related work as only 10.7 % of the respondents have more than seven years of experience with re-engineering using MSA technologies. Most of the respondents (78.6 %) have one to six years of experience related to MSA re-engineering, Table 2.

Table 2
How many years of experience you have on working with microservices, or re-engineering projects involving migrating to microservices?

Choice	#	%
Less than 1 year	3	10.7%
1-3 years	9	32.2%
4-6 years	13	46.4%
7-10 years	2	7.1%
More than 10 years	1	3.6%

Regarding the roles of the respondents. We can observe that most are working in a senior position, with the most common roles being senior developer (42.9 %) and architect (21.4 %). 82.1 % of the respondents work in roles that can be described as senior (senior developer, architect, project manager, higher management, product owner). Only 14.8 % of the respondents identify as junior developers, Table 3.

The most common business domain for the software products our respondents worked on was banking or accounting-related software (18.5 %). Other popular domains were industrial manufacturing and retail markets, both with 11.1 %. There is clear scattering in this question, as 44.5 % of the respondents didn't identify with any of the choices, Table 4.

Table 3
Which of these titles would characterize your role in your most recent MSA-project?

Choice	#	%
Junior Developer	4	14.3%
Senior Developer	12	42.9%
Architect	6	21.4%
Domain expert	0	0.0%
Product owner	1	3.6%
Project manager / Team leader	2	7.1%
QA, or other, manager	0	0.0%
Higher management or administration	2	7.1%

Table 4
What was the business domain of the software product in your most recent MSA project?

Choice	#	%
Resource production (farming, forest ind.)	1	3.7%
Industrial manufacturing	3	11.1%
Banking or accounting	5	18.5%
Retail market, online or store-based	3	11.1%
Logistics, or shipping industry	2	7.4%
Energy, heat or water distribution	1	3.7%
Automotive Industry	0	0.0%
Other	12	44.5%

In the size of a typical MSA project team, the projects usually include between 6-10 (46.4 %) or 3-5 (28.6 %) personnel. Most respondents worked in teams with 3-10 team members (75 %). Small (less than 3) or large teams (11-25 or more) are less common, Table 5.

Table 5
In average, how many people are involved your organization's typical MSA project?

Choice	#	%
Less than 3	2	7.1%
3-5	8	28.6%
6-10	13	46.4%
11-25	1	3.6%
more than 25	4	14.3%

93 % of the respondents report changes in the project personnel count; sometimes (53.6 %), often (21.4 %), rarely (14.3 %) or always (3.6 %). Nearly all of the MSA projects have some amount of fluctuation in the personnel count. However, the bulk of the fluctuation happens only sometimes, Table 6.

Table 6
Does the amount of people involved change, or fluctuate during the MSA project?

Choice	#	%
Never	2	7.1%
Rarely	4	14.3%
Sometimes	15	53.6%
Often	6	21.4%
Always	1	3.6%

The most common team size in MSA projects is one team, which accounts for 40.8 % of projects. The second most common team size is 2-3 teams in almost the same number of projects. Less commonly, 4-5 teams are used in 14.8 % of projects, while projects with more than 5 are the least popular, accounting for only 7.4 % of projects, Table 7.

Table 7
In average, to how many teams are these people typically divided into?

Choice	#	%
Just one	11	40.8%
2-3	10	37.0%
4-5	4	14.8%
more than 5	2	7.4%

The survey revealed that the number of teams in a given MSA project rarely changes (50 %), sometimes changes (35.7 %), never changes (7.1 %), or changes often (3.6 %), and always (3.6 %), respectively, Table 8.

Table 8
Does the amount of teams involved change or fluctuate during the MSA project?

Choice	#	%
Never	2	7.1%
Rarely	14	50.0%
Sometimes	10	35.7%
Often	1	3.6%
Always	1	3.6%

The majority of the legacy systems that MSA is replacing are based on monolithic architecture (81.5 %), followed by client-server (7.4 %) and service-oriented architectures (7.4 %). Only one project used other solutions and designs (3.7 %), Table 9.

The data shows that most systems being replaced are between 7 and 10 years old, making up 37.1 % of the

Table 9
In projects where microservice solutions are developed to replace existing systems, are those systems typically based on

Choice	#	%
Monolithic design	22	81.5%
Client-server design	2	7.4%
Service-Oriented design	2	7.4%
Peer-to-Peer design	0	0.0%
Layered design	0	0.0%
Other solutions /architectures	1	3.7%

total. The next highest age group is 11 to 15 years old, representing 25.9 % of the total. Other age groups include systems aged 4 to 6 years and 16 to 20 years, each making up 14.8 % of the total. Systems that are less than 3 years old and older than 20 years each account for 3.7 % of the total.

Based on the results, most microservice projects are being implemented to replace systems between 7 and 15 years old (63 %), with fewer systems falling outside of this range. Interestingly, the most common age range for replacement is between 7 and 10, Table 10.

Table 10
In projects where microservices solutions to replace existing systems are development, how old in general are those systems that are being replaced?

Choice	#	%
Less than 3 year	1	3.7%
4-6 years	4	14.8%
7-10 years	10	37.1%
11-15 years	7	25.9%
16-20 years	4	14.8%
Older than 20 years	1	3.7%

Most Microservices Architecture (MSA) projects have a duration range of 1-3 years, representing 35.7 % of the total. Following this, projects lasting 7-12 months account for 32.2 % of the total. Projects with less than 3 months and 4-6 months represent smaller proportions of the total, at 7.1 % and 10.7 %, respectively. A small percentage of projects have longer duration, with 3.6 % lasting between 4-5 years and 10.7 % lasting more than 5 years. MSA projects usually span 1-3 years, with a significant proportion lasting 7-12 months, Table 11.

MSA projects typically involve the development of 2-10 self-contained microservices, with 35.7 % consisting of 2-5 services and 25.0 % consisting of 6-10 services. Additionally, 21.4 % of projects involve developing 10-25 services. Fewer projects have a smaller or larger number of microservices, with only 10.7 % involving one service and 3.6 % each involving 26-50 services or more than fifty services. MSA projects usually involve developing a moderate number of self-contained microservices, ranging from a few services to several tens of services, Table 12.

Table 11
In general, what is the duration of your organization's typical MSA project?

Choice	#	%
Less than 3 months	2	7.1%
4-6 months	3	10.7%
7-12 months	9	32.2%
1-3 years	10	35.7%
4-5 years	1	3.6%
More than 5 years	3	10.7%

Table 12
In general, how many different self-contained microservices are developed for a one typical MSA project?

Choice	#	%
One service	3	10.7%
2-5 services	10	35.7%
6-10 services	7	25.0%
10-25 services	6	21.4%
26-50 services	1	3.6%
More than fifty services	1	3.6%

4.2. Success factors and methodologies

In this section, we discuss the results related to the success factors and methodologies utilized by the respondents. When asked, "In a few words, can you please mention three of the most important aspects that, in your opinion, help MSA-related projects to be successful?" the respondents highlighted several crucial aspects essential for the success of projects related to Microservices Architecture (MSA).

Firstly, adopting a cross-functional approach, enabled by MSA, allows teams to work independently on the development, testing, troubleshooting, deployment, and updating of services. This approach results in faster deployment and troubleshooting. It emphasizes the importance of planning and adopting a DevOps methodology to streamline the development and operations processes.

Secondly, careful planning and considering architectural trade-offs are essential to avoid common pitfalls like the distributed monolith trap. This involves understanding the implications of MSA and making informed decisions regarding technology choices and service boundaries.

Thirdly, effective team and stakeholder communication ensures alignment and collaboration throughout the migration process. Clear communication facilitates knowledge sharing, coordination, and support from management, which are critical for project success.

Other aspects highlighted include the importance of proper documentation, developer skills, infrastructure automation, well-defined service boundaries, and early system performance testing and investigation. Additionally, management buy-in, architectural design, and deployment strategies contribute to the overall success of MSA-related projects.

In summary, successful MSA-related projects require a holistic approach encompassing technical expertise, effective communication, careful planning, and continuous testing and refinement to address the challenges of migrating to Microservices Architecture.

In MSA migration projects, service boundaries are typically derived through intuition and skills based on previous experience, which accounts for 46.2 % of the total. Another significant proportion of projects (42.3 %) uses the approach of dividing services based on domain boundaries, often associated with Domain-Driven Design (DDD) principles. Only a few projects (3.8 %) use analysis tools to derive service boundaries, while a small percentage of projects (7.7 %) utilize other unspecified methods, Table 13.

Table 13
In your typical MSA migration project, how do you derive service boundaries?

Choice	#	%
Through dividing services based on domain boundaries (DDD).	11	42.3%
Through analysis tools.	1	3.8%
Through intuition and skills based on earlier experience.	12	46.2%
Other	2	7.7%

In most MSA migration projects, diagram modeling tools like UML, draw.io, Miro, or Visio document the new system, accounting for 55.6 % of the total. 18.5 % of projects document the system with separate documents.

On the other hand, textual modeling tools or documentation in code comments are used in fewer projects, representing only 3.7 % each. Similarly, a small percentage of projects (3.7 %) do not systematically document their projects, while 14.8 % employ other approaches not specified in the predefined categories, Table 14.

In the survey, 25 % of respondents considered the source code of the previous system the most crucial data source for planning the migration to a Microservices Architecture (MSA). Existing documentation in diagrams (e.g., UML diagrams) was selected by 22.2 % of the respondents, followed by the data schema, chosen by 18.1 % of the respondents. Other data sources were also considered necessary, with smaller proportions of respondents selecting them, ranging from 1.4 % to 11.1 %, Table 15.

Table 14
In your typical MSA migration project, how do you document the new MSA-based system?

Choice	#	%
With diagram modelling tool (for example UML, draw.io, Miro, Visio).	15	55.6%
With textual modelling tools (for example Mermaid).	1	3.7%
With documentations in comments in code.	1	3.7%
With documentation in separate documents.	5	18.5%
We do not systematically document our projects.	1	3.7%
With other approach.	4	14.8%

Table 15
Select up to three most important data sources, which you considered critical on planning the migration process

Choice	#	%
Source code of the previous system.	18	25 %
Existing documentation in diagrams (for example UML-diagrams).	16	22.2%
Documentation in text or source code comments.	5	6.9%
Existing tests and test automation cases.	8	11.1%
Data schema	13	18.1%
Unwritten knowledge caught with interviews.	8	11.1%
Data from analysis tools.	1	1.4%
Other	3	4.2%

4.3. Challenges and motivations

In this section, we go through the challenges and motivations of the migration process utilizing the horseshoe model. According to a survey, the most common motivation for implementing a microservices architecture (MSA) in migration projects was to achieve a more cohesive and less coupled project, with 33.4 % of respondents selecting this as their primary motivation. This highlights the importance of maintainability. Around 14.8 % of respondents cited various motivations, such as gaining scalability through multiple microservices, allowing for partial upgrades of the system, and opting for a part-by-part migration from the legacy system. Similarly, 11.1 % opted to enable teams to work more independently. Additionally, 7.4 % of respondents chose other motivations that were not explicitly specified. Interestingly, none of the respondents selected decreased time to market as a primary motivation for adopting MSA in their migration projects, Table 16.

Table 16
On the last migration project from old system to MSA, what was the main motivation for this migration process to take place?

Choice	#	%
To gain scalability through multiple microservices.	4	14.8%
To gain more cohesive and less coupled project (maintainability).	9	33.4%
Decreased time to market.	0	0.0%
Faster development iterations.	1	3.7%
Enable teams to work more independently.	3	11.1%
Allow for partial upgrades of your system.	4	14.8%
Part by part migration from legacy system.	4	14.8%
Other	2	7.4%

Various challenges were faced during the planning phase of migrating to MSA. One of the major obstacles identified by a significant proportion of respondents (18.1 %) was the lack of documentation. Furthermore, other common issues encountered include lack of test coverage (13.6 %), tightly coupled components (11.8 %), obscure boundaries between components and functions (11.8 %), and missing knowledge of the code base (11.8 %). However, fewer respondents reported unexpected side effects in components (8.2 %) and technical issues (7.3 %), Table 17.

Several challenges arise during the restructuring phase of system migration, as highlighted by the survey respondents. The decomposition of the existing system is a significant obstacle reported by approximately 18.8 % of the respondents. This process involves breaking down the system into smaller, more manageable components or services, requiring careful consideration to ensure smooth transitions and maintain system functionality.

Furthermore, approximately 16.3 % of the respondents cited the lack of documentation as a challenge during restructuring. Adequate documentation is crucial for understanding the system's architecture and dependencies, facilitating effective decision-making and communication among team members.

Additionally, the respondents identified challenges related to reducing coupling in the new system (15 %), determining the appropriate granularity for microservices (15 %), managing tightly coupled components (11.3 %), and recognizing microservice boundaries (11.3 %). Other challenges mentioned by respondents included the lack of automated tests (7.5 %), lack of code comments (3.8 %), and unspecified issues (1.3 %), Table 18.

Table 17
When planning a migration, it is necessary to reverse engineer (understand and abstract) the existing system. What were the challenges when planning the migration to MSA?

Choice	#	%
Tightly coupled components.	13	11.8%
Unexpected side effects in components.	9	8.2%
Lack of test coverage.	15	13.6%
Lack of documentation.	20	18.1%
Missing knowledge of code base.	13	11.8%
Obscure boundaries between components and functions.	13	11.8%
Lack of confidence in the code base.	6	5.5%
Convincing management or other parties about the migration.	8	7.3%
Informing related stakeholders.	3	2.7%
Technical issues.	8	7.3%
Other.	2	1.8%

Table 18
When transforming a system, it is necessary to restructure (extend and improve) the existing system. What were the challenges when restructuring the system during the migration process?

Choice	#	%
Tightly coupled components.	9	11.3%
Difficulty recognizing microservice boundaries.	9	11.3%
Decomposition of the existing system.	15	18.8%
Lack of automated tests.	6	7.5%
Reducing coupling in the new system.	12	15%
Figuring out the right granularity for microservices.	12	15%
Lack of documentation.	13	16.3%
Lack of code comments.	3	3.8%
Other	1	1.3%

The migration to MSA can present various challenges when forward engineering a system. According to a survey, approximately 14.4 % of respondents identified challenges related to adapting to new development methodologies necessitated by microservices, and 13.4 % established the infrastructure required for microservices.

Communication and coordination between teams emerged as another prominent challenge, with approximately 13.4 % of respondents citing this as an obstacle. Effective communication is essential for ensuring alignment and collaboration among teams working on different aspects of the microservices-based system.

In addition, respondents noted challenges related to monitoring and debugging microservices-based systems, with approximately 14.4 % and 12.4 %, respectively, reporting difficulties in these areas. Smaller percentages of respondents also cited challenges related to monitoring microservice logging and standardizing microservices. Testing the new microservices-based system presented challenges for approximately 9.3 % of respondents. Other challenges mentioned by respondents included building the first proof of concept (3.1 %) and unspecified issues (4.1 %), Table 19.

Table 19
When forward engineering a system, it is necessary to generate (refine and improve) code. What were the challenges when forward engineering the system during the migration process?

Choice	#	%
Establishing the infrastructure required for microservices.	13	13.4%
New development methodology because of microservices.	14	14.4%
Monitoring microservice-based system.	14	14.4%
Communication between teams.	13	13.4%
Monitoring microservice logging.	7	7.2%
Debugging microservice system.	12	12.4%
Standardizing microservices.	8	8.2%
Testing new microservice-based system.	9	9.3%
Building the first proof of concept (PoC).	3	3.1%
Other	4	4.1%

We got varied responses when asking for open comments about MSA migration at the end of the survey. The comments warn about the complexity of MSA migration and stress the importance of careful planning and refactoring to avoid issues such as the distributed monolith trap. Financial implications are mentioned. Security concerns and the significance of data transfer between services are also highlighted. Furthermore, experiences with both successful and unsuccessful MSA implementations emphasize the need for thorough planning and execution.

5. Discussion

The first research question focused on the project and demographic details related to MSA migrations. With this research question, we wanted to know what kind of organizations, teams, and timelines are involved in migrating toward MSA. The results regarding the first research question show that most respondents have significant software development backgrounds, indicating that experienced professionals usually undertake MSA migrations. 60.7 % of participants have over ten years of experience, while 89 % have at least four years, highlighting the advanced expertise required for successful MSA migration projects. A significant 77.8 % of respondents have between one to six years of experience, specifically in MSA re-engineering. This suggests that while MSA is a relatively newer field, many professionals have quickly gained considerable expertise.

Regarding the company domain, the banking or accounting sector seems to be the most common field for MSA projects, emphasizing the critical need for scalable and flexible architectures in financial services. However, there is a lot of scattering in this question, with 42.3 % not identifying with any specific domain, illustrating the widespread applicability of MSA across various sectors. Teams usually have 3-10 members, suggesting that medium-sized teams are optimal for MSA projects. However, team size fluctuation is typical, with nearly all projects experiencing some degree of fluctuation (93 %), primarily occurring only sometimes (53 %). This likely reflects these projects' adaptive and iterative nature.

Most MSA projects (81.5 %) are aimed at replacing outdated monolithic systems, with these legacy systems being between 7 and 15 years old. This indicates a strong trend towards modernization, with MSA offering a viable pathway out of the limitations of older architectures.

The duration of MSA projects varies, with a notable portion taking between 1-3 years (35.7 %) or 7-12 months (32.2 %). This range reflects both the complexity and the scale of such migrations. A vast majority deploy a moderate number of microservices (2-25), with the most common range being 2-5 microservices (35.7 %). This suggests a cautious approach to microservices deployment, likely aimed at managing complexity and ensuring stability during the transition.

The second research question targeted the methodologies and success factors of the MSA project. We wanted to know what methodologies were used in MSA migration-related tasks and what critical factors could aid in MSA migration. The result of the second research question shows that in MSA projects when determining service boundaries, 46.2 % of respondents rely on intuition and skills from previous experience to assess service boundaries. In comparison, 42.3 % use DDD principles. This indicates a balanced reliance on empirical knowledge and methodical, principle-based strategies.

Regarding documentation practices, 55.6 % of respondents indicated using diagram modeling tools such as UML, draw.io, Miro, or Visio in documenting MSA migrations, suggesting a strong preference for a visual representation of system architecture. Meanwhile, 18.5 %

rely on separate documents for documentation, possibly indicating a supplementary strategy to detailed diagrams or a preference for textual over visual documentation in specific contexts.

When asked about the sources of information for planning migrations, 25 % of respondents consider the source code of the previous system as the most crucial data source for migrating to MSA, while 22.2 % view existing documentation, such as diagrams (e.g., UML), as the most valuable.

These findings highlight the importance of a balanced approach that values both experience and theoretical frameworks in defining service boundaries, the widespread use of visual documentation in understanding, planning, and communicating the architecture of MSA systems, and the necessity of a deep understanding of the legacy system and the importance of thorough documentation practices.

The third research question was aimed at understanding the motivations and challenges of the migration process. Through these questions, we wanted to comprehend what motivates migrations, the challenges, and at what phase of the migration challenges arise (mapped using the horseshoe model). Results of the third research question show that the main reason for adopting MSA, as reported by 33.4 % of respondents, is to create a more cohesive and less coupled project structure. This strategic goal aims to improve system modularity, making it easier to maintain, evolve, and scale applications. Other significant motivations include the ability to scale parts of the system independently through multiple microservices and the flexibility to perform partial system upgrades. Allowing the teams to work more independently is also a key driver, leading to organizational benefits such as enhanced productivity and faster development cycles. This aligns with the MSA principle of giving teams autonomy over their respective services, which can improve innovation and efficiency.

During the reverse engineering stage of the migration process, the most prominent challenges include a lack of documentation, insufficient test coverage, and obscure boundaries between components and functions. These issues underscore the difficulties in understanding and preparing the existing system for a transition to MSA, emphasizing the need for thorough groundwork and clear system delineation.

Challenges such as the decomposition process, managing tightly coupled components, and a lack of documentation emerge during the transformation phase of the migration process. These reflect the technical complexities of breaking down a monolithic system into microservices, which requires a deep understanding of the domain and the existing system's architecture.

The results from the forward engineering phase of MSA migration reveal several key challenges that organizations encounter as they transition to this architectural style. These challenges included technical and infrastructural hurdles and underscored the significance of organizational and process-oriented adjustments.

Regarding participant demographics, our research aligns with Taibi et al. [13] and Francesco et al. [15]. Our results show that most respondents have significant experience in software development, with a substantial portion specifically experienced in MSA re-engineering. This emphasizes the importance of experienced personnel in MSA projects.

Furthermore, we observed small to medium team sizes, which reflected the industry standard. It is a testament to MSA's encouragement of smaller, more agile teams, a point also mentioned by Francesco et al. [15] and Taibi et al. [13]. Our specific findings on the number of microservices deployed and the project durations provide detailed insights into the scale and time frame of MSA migrations, which are less explicitly detailed in the comparative studies. Interestingly, our results show the generally positive management reaction to MSA migration and the high degree of business-IT alignment, which the comparative literature does not cover extensively.

We observed cohesive and less coupled projects as a primary motivation for MSA migration. Authors like Taibi et al. [13] and Fritzsch et al. [16] also highlight motivations such as scalability, maintainability, and the modular architecture of microservices as drivers for adoption.

Regarding the challenges in the reverse engineering phase, we have identified several challenges, including the lack of documentation, insufficient test coverage, and tightly coupled components. Similar findings were reported by Francesco et al. [15]. However, their participants were more concerned with time to market, high amount of coupling, and maintenance difficulties. Challenges related to understanding the system, documenting it, and identifying independent components for decoupling are recurring themes, emphasizing the crucial need for clear system understanding and abstraction capabilities.

We have identified several critical challenges related to the transformation phase, including the system's decomposition, lack of documentation, tight coupling, microservice granularity, and boundary recognition. In their survey, Francesco et al. [15] also found similar challenges, emphasizing the nuanced difficulties of restructuring systems for MSA migration. These results show the multifaceted nature of MSA migration, encompassing technical, organizational, and documentation-related challenges in system transformation.

In the forward engineering phase, the most common challenges included adopting a new development methodology, setting up infrastructure, monitoring the MSA system, communicating between teams, and debugging the

MSA system. These challenges emphasize the need to adopt new methods, set up infrastructure, and address operational challenges such as monitoring, debugging, and ensuring effective communication. Francesco et al.'s [15] findings echo many of these concerns, highlighting the need for a mindset shift among developers and the overarching goal of achieving uniformity across services. These insights underscore the multifaceted nature of adopting MSA, encompassing initial planning, restructuring, practical execution, and standardization of the new architecture.

6. Threats to validity

We used the classification by Wohlin et al. [19] to classify the threats to validity related to this research paper. Wohlin et al. divide threats to validity into four classes: conclusion, internal, external, and construct. We have identified internal, external, and conclusion threats. Internal threats affect the study results without the knowledge of the researchers. External threats hinder the application of the results to the real world. Conclusion threats affect the credibility of the conclusion based on the results.

Regarding the internal bias, we identified a threat in the control of respondents. We did not have control over the respondents as the survey was conducted online. To mitigate this threat, we attempted to contact people in person to confirm their skill set before asking them to complete the survey, and we marketed the survey on social media in specialized forums to get the people with the right expertise to fill it. Additionally, we read through the responses carefully to identify any misleading or harmful responses. We found none of them.

Another form of internal bias is researcher bias, often introduced to the research process by researchers and their conscious and unconscious beliefs and expectations. To mitigate this bias, we will be transparent by publishing everything related to this study, including the survey form and the results, so that other researchers can verify our conclusions.

Regarding the external threats to validity. We identified the low rate of respondents as a possible threat. 339 respondents opened this survey; it was started by 66 and submitted by 28 persons, giving us a response rate of 8.2 %, which is less than one in ten but still very good for an online survey. However, it has to be remembered that due to difficulties in identifying a large enough audience with relevant skills, this survey was advertised on social media and websites aimed at professional software developers. Furthermore, low respondent numbers are expected when surveying such a specialized field. In order to give perspective, according to the Finnish Ministry of Economic Affairs, the whole of the Finnish software development sector consisted of 53000 personnel in 2018 [20]. We could not find data to estimate how many people work with MSA migrations.

Concerning the conclusion bias, we identified the possibility of sampling bias, meaning that the population surveyed does not represent the entire population. This is a concern as we did not have many respondents. However, we consider the number of respondents enough for this study as the pool of possible candidates in Finland is not that large.

7. Conclusion

Software life-cycle means all products need replacement, revision, or removal from the corporate portfolio. However, some systems deemed valuable enough are replaced with modernized versions of the same system, usually meaning that they are migrated from monolithic systems to microservice architectures. In this paper, we surveyed 28 industry professionals to gather project details, motivation, and challenges of MSA migration.

Our survey results indicated that most respondents had significant experience, suggesting that experienced practitioners mainly do MSA migration. The seniority of their roles supports this claim. Banking or accounting is the most cited business domain, though a significant portion did not specify a domain. Most work in teams of 3-10 members, aligning with industry norms despite the notable presence of larger teams. MSA projects frequently experience team size fluctuations. Service boundaries are often derived using intuition or previous experience, with many also using DDD principles. The primary motivation for MSA migration is to achieve a more cohesive, less coupled project, underlining the value of maintainability. The main challenges in reverse engineering of the migration process include a lack of documentation, insufficient test coverage, component coupling, and vague component boundaries. In the transformation phase, the main challenges are decomposing existing systems, lack of documentation, reducing coupling, and determining microservice granularity. Finally, the challenges in the forward engineering phase were adapting to new methodologies, establishing infrastructure, team communication, system monitoring, and debugging.

Based on the literature review of similar works, we can observe common motivations such as scalability and maintainability and challenges like decoupling from monoliths, migrating databases, and establishing effective microservice communication. Key findings across studies also stress the need for new development methodologies and addressing infrastructure requirements as critical aspects of MSA migration. Our research adds to the body of knowledge by introducing insights into adapting to microservices-specific practices and the operational adjustments necessary for successful migration. Moreover, our study complements existing findings by broadening the geographic scope of MSA migration research. It provides a broader perspective on the challenges and motivations for adopting MSA across different regions.

For future work, we aim to move forward with our observations and conduct qualitative studies in selected software organizations to understand better the potential pitfalls and requirements for successful migration processes. With our observations, we intend to define a framework that could be applied in the software maintenance and re-engineering life cycles to prepare legacy systems for migration and modernization processes or assess the feasibility of committing to one as a replacement for a legacy system.

References

[1] D. L. Parnas, Software aging (1994). doi:10.1109/ICSE.1994.296790.

[2] A. Balalaie, A. Heydarnoori, P. Jamshidi, Microservices architecture enables devops: Migration to a cloud-native architecture, IEEE Software 33 (2016) 42–52. doi:10.1109/MS.2016.64.

[3] M. Garriga, Towards a taxonomy of microservices architectures, in: In: Cerone A., Roveri M. (eds) Software Engineering and Formal Methods. SEFM 2017. Lecture Notes in Computer Science., volume 10729, Springer, 2018, pp. 203–218. doi:10.1007/978-3-319-74781-1_15.

[4] M. Richards, Microservices vs. service-oriented architecture, in: Computer Science - Research and Development, O'Reilly Media, 2016.

[5] F. M, Microservices Guide, Technical Report, martinfowler.com, 2014.

[6] P. Calcado, Building Products at SoundCloud—Part II: Breaking the Monolith, Technical Report, SoundCloud, 2014.

[7] C. Bampis, C. Chen, A. Moorthy, Z. Li, Netflix Video Quality at Scale with Cosmos Microservices., Technical Report, Medium, 2021.

[8] E. Haddad, Service-Oriented Architecture: Scaling the Uber Engineering Codebase As We Grow., Technical Report, Uber, 2015.

[9] D. S. Linthicum, Practical use of microservices in moving workloads to the cloud, IEEE Cloud Computing 3 (2016) 6–9.

[10] W. Hasselbring, G. Steinacker, Microservice architectures for scalability, agility and reliability in e-commerce, in: 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), IEEE, 2017. doi:10.1109/ICSAW.2017.11.

[11] K. Tuusjärvi, J. Kasurinen, S. Hyrynsalmi, Migrating a legacy system to a microservice architecture, e-Informatica Software Engineering Journal 18 (2024) 240104. doi:10.37190/e-Inf240104.

[12] R. Kazman, S. Woods, S. Carriere, Requirements for integrating software architecture and reengineering models: CORUM II, in: Proceedings Fifth Working Conference on Reverse Engineering (Cat. No.98TB100261), 1998, pp. 154–163. doi:10.1109/WCRE.1998.723185.

[13] D. Taibi, V. Lenarduzzi, C. Pahl, Processes, motivations and issues for migrating to microservices architectures: An empirical investigation 4 (2017). doi:10.1109/MCC.2017.4250931.

[14] J. Ghofrani, D. LÃ¼bke, Challenges of microservices architecture: A survey on the state of the practice, in: ZEUS, 2018.

[15] P. Francesco, P. Lago, I. Malavolta, Migrating towards microservice architectures: An industrial survey, 2018, pp. 29–2909. doi:10.1109/ICSA.2018.00012.

[16] J. Fritzsch, J. Bogner, S. Wagner, 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. doi:10.1109/ICSME.2019.00081, ISSN: 2576-3148.

[17] J. Bogner, J. Fritzsch, S. Wagner, A. Zimmermann, Microservices in industry: Insights into technologies, characteristics, and software quality, in: 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), 2019, pp. 187–195. doi:10.1109/ICSA-C.2019.00041.

[18] H. Knoche, W. Hasselbring, Drivers and barriers for microservice adoption â a survey among professionals in germany 14 (2019) 1:1–35. doi:10.18417/emisa.14.1.

[19] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in Software Engineering, Springer, 2012. doi:10.1007/978-3-642-29044-2.

[20] Ministry of Economic Affairs, Toimialaraportit. ohjelmistoala 2020 (2020) 24–25.