Migrating towards Microservice Architectures: an Industrial Survey

Paolo Di Francesco
Gran Sasso Science Institute
L'Aquila, Italy
e-mail: paolo.difrancesco@gssi.it

Patricia Lago, Ivano Malavolta
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
e-mail: p.lago@vu.nl, i.malavolta@vu.nl

***Abstract*—Microservices are gaining tremendous traction in industry and a growing scientific interest in academia. More and more companies are adopting this architectural style for modernizing their products and taking advantage of its promised benefits (e.g., agility, scalability). Unfortunately, the process of moving towards a microservice-based architecture is anything but easy, as there are plenty of challenges to address from both technical and organizational perspectives.**

In this paper we report about an empirical study on migration practices towards the adoption of microservices in industry. Specifically, we designed and conducted a survey targeting practitioners involved in the process of migrating their applications and we collected information (by means of interviews and questionnaires) on (i) the performed activities, and (ii) the challenges faced during the migration. Our findings benefit both (i) researchers by highlighting future directions for industry-relevant problems and (ii) practitioners by providing a reference framework for their (future) migrations towards microservices.

***Index Terms*—Microservice Architecture, Migration to Microservices, Industrial Survey**

I. Introduction

There is no single and rigorous definition about the microservice architecture (MSA) style. The most adopted definition in scientific papers [5], is the one provided by Lewis and Fowler, which describes this style as *an approach for developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API* [8].

The MSA style brings both significant benefits and challenges. If on one hand microservice systems have flexible and evolvable architectures [6], on the other hand many technical challenges (e.g., infrastructure automation, distributed debugging) and organizational challenges (e.g., creation of cross-functional teams) need to be faced before being able to fully benefit from them. Adopting MSAs is challenging [20], either for developing a greenfield system, or for migrating an existing system that suffers from issues that become more and more difficult to solve. Typical issues of legacy systems are technical (e.g., the system becomes highly coupled, hard to maintain, presents side effects) or business-related (e.g., long time to release new features, low productivity of developers). In some cases, migrating towards MSA represents the best option for resolving existing issues and at the same time improving the system maintainability and the frequency of product releases.

In this study we present an in-depth investigation among practitioners to uncover the migration practices towards MSA. To this aim, we study (i) the activities and (ii) the challenges faced by industrial practitioners when migrating towards MSA. The research **methodology** we used relied on two main phases. In the first phase, we designed an interview guide that we used for conducting 5 exploratory interviews with industrial practitioners. In the second phase, we used the interview guide and the results of the interviews to design an online survey questionnaire, which we distributed among our network of practitioners. In total, 18 practitioners across 16 different IT companies and at different professional stages participated to our study. The high-quality data contributed by our participants makes this a rather exciting first study in the state of the practice in migrations towards MSA, and hence an hopefully inspiring food for thought for future research.

The main **contributions** of this paper are the following:

- a survey of 18 practitioners that provides quantitative information about migrations towards MSA;
- an analysis of the collected data that discusses practitioners perspectives on migration activities and challenges in each of the three phases of: (i) architecture recovery of the pre-existing system, (ii) architecture transformation from the pre-existing to the new architecture, and (iii) the implementation of the new system;
- a discussion of the obtained results in terms of practitioners activities and potentially relevant research directions;
- the study replication package[1], featuring the raw data of the online questionnaire.

The rest of the paper is organized as follows. In Section II we describe the migration model for MSA we used throughout the study. In Section III we present the design of the study. In Sections IV, V, and VI we present the results of our survey, followed by our reflections on the topic of migrations towards MSA (Section VII). Threats to validity and related work are described in Sections VIII and IX, respectively. With Section X we close the paper and discuss future work.

II. Migrating towards MSA

Inspired by the *horseshoe model* by Kazman et al. [13], Razavian et al. [23] framed the processes characterizing the

[1]http://www.s2group.cs.vu.nl/icsa-2018-replication-package

978-1-5386-6398-1/18/$31.00 ©2018 IEEE
DOI 10.1109/ICSA.2018.00012

Authorized licensed use limited to: University of L'Aquila. Downloaded on September 08,2021 at 14:30:21 UTC from IEEE Xplore. Restrictions apply.

migration to services into three steps: reverse engineering, architecture transformation, and forward engineering (thick arrows in Figure 1). For this study, we applied the same characterization to the process of migrating towards MSA.

As any migration is motivated by the need to modernize a system, in part or in full, to some extent such system is considered a legacy. In our work, we refer to the system existing before the migration as the *pre-existing system* (i.e., the legacy asset), while we refer to the targeted microservice system as the *new system*.

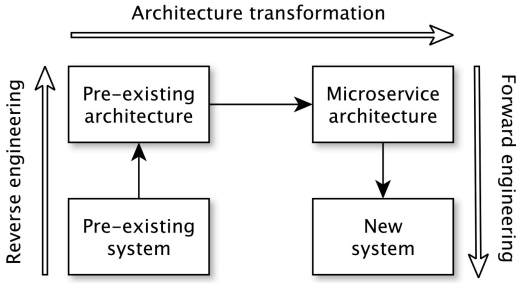

Fig. 1. Migration to microservices (adapted from Kazman et al. [13])

In the *reverse engineering* step, architects consider and analyse the pre-existing system (e.g., by means of code analysis tools or some existing documentation) and identify the legacy elements which are candidates for transformation to services. Then, the *transformation* step involves the restructuring of the pre-existing architecture into a microservice-based one (still at the same level of abstraction), for example by reshaping design elements, restructuring the architecture and altering business models and business strategies. Finally, in the *forward engineering* step, the design of the new system is finalized, implemented, and deployed (possibly in production).

III. Study Design

In this section we describe the design of the study in terms of goal and research questions, target population and sampling, design of the interviews and questionnaire, and data analysis.

A. Goal and Research Questions

The main **goal** of this study is to characterize the activities and the challenges faced by industrial practitioners when migrating towards MSA. We refined our goal in the following two research questions.

RQ1 – *What are the activities carried out by practitioners when migrating towards a microservice-based architecture?* By answering this research question we aim at characterizing the activities performed during the overall migration process.

RQ2 – *What are the challenges faced by practitioners when migrating towards a microservice-based architecture?* By answering this research question we aim at characterizing the challenges that practitioners have to face during the migration.

B. Identification of the Target Population

The target population for this survey entails industrial practitioners that were involved in at least one migration of a pre-existing system towards MSA. We included also practitioners that were in the process of planning a migration (i.e., when the actual implementation of the migration was not started yet). We identified the initial group of practitioners in our network of industrial collaborators. We contacted them directly and we began the exploratory interviews.

We extended our network of practitioners by asking the participants to the interviews, to nominate additional experts in their own networks (applying the *snowballing sampling approach* [15]). We also included (i) mailing lists of software architects, (ii) IT consultancy companies, and (iii) industrial authors that have published articles on the topic of microservices. The resulting population was invited for the second phase, the online questionnaire survey.

C. Design of the Questionnaire

In this research we follow the well-established guidelines for interviews and questionnaire design by Oppenheim [21]. We applied a combination of both interviews and questionnaires.

At the beginning of our investigation, we carefully designed an interview guide used to conduct 5 exploratory interviews with industrial practitioners with experience in the migration towards MSA. The duration of the interviews ranged from 30 to 80 minutes. The interviewee was provided upfront with the same horseshoe model shown in Figure 1 and a brief description of its various phases. We also asked the interviewee to think of a specific migration project he or she is or was involved in. Each interview was structured as follows. The interviewer started by introducing the participant with the goal of the survey. The interviewer then asked some demographic questions, followed by general questions on the migration (e.g., domain of the application, number of teams). Afterwards, three sets of specific questions were asked, one for each sub-process of the *horseshoe model* described in Section II, respectively addressing the following aspects: (i) architecture recovery of the pre-existing system, (ii) architecture transformation from the pre-existing to the new architecture, and (iii) architecture-based development (i.e. the implementation of the new system). Before concluding the interview, the participant was always asked for comments and remarks about relevant aspects of the migration that were possibly left uncovered. The interviewer kept the minutes of each interview, used afterwards for designing the questionnaire.

The questionnaire is composed of 35 questions. All mandatory questions are closed-ended. In 4 optional questions, the participant was asked to add details and comments. The structure of the questionnaire follows the structure of the interview. Beside the introduction and some preliminary definitions, the core parts of the questionnaire are: (i) demographic questions, (ii) general questions on the system, (iii) general questions on the migration, (iv) general questions on the organization, (v) questions about the recovery of the pre-existing system, (vi) questions about the architecture transformation, (vii) questions about the new system implementation, and (viii) questions about the challenges. Parts (i-iv) of the questionnaire are used to collect demographic data about the migration. The remaining parts (v-viii) are used to collect data

Authorized licensed use limited to: University of L'Aquila. Downloaded on September 08,2021 at 14:30:21 UTC from IEEE Xplore. Restrictions apply.

about the migration activities and the related challenges. The questionnaire was created as an online form, and an invitation was sent by email to the target population (see Section III-B).

D. Data Analysis

We analyzed the data according to the following steps.

1. Coding of interview notes. The main goal of this step is to categorize the data obtained during the interviews in order to make it homogeneous with respect to the structure of the questionnaire. Specifically, we considered the detailed notes collected during the 5 interviews and categorized them according to the structure of the online questionnaire. In this step we applied the closed coding technique, where codes were pre-determined based on the specific questions and options within the questionnaire [18]. It is important to note that this step has been strongly eased by the fact that the data coming from the interviews and the online questionnaire were already aligned since the structure of the online questionnaire was derived from the interview guide. In this step we also collected all the segments of the interview notes that did not fit into the structure of the questionnaire, leaving them for further analysis. This eliminated the possibility that we accidentally skipped any segment of the interviews.

2. Collection of questionnaire responses. We collected all the answers provided by the participants of the online questionnaire into a single spreadsheet (with a column for each question and a row for each participant).

3. Data analysis. This step takes as input a fully populated spreadsheet with all the information coming from both interviews and the online questionnaire. So, for each question (i.e., for each column of the spreadsheet) we applied basic descriptive statistics for better understanding the data about the occurrences of each given response. The contents of the 4 optional questions have been used for understanding better the rationale and getting additional information about their corresponding mandatory questions (see next step).

4. Findings discovery. In this step we collected the results of all the statistical analyses, we extracted key findings and we aggregated them into a set of bar plots and tables. Then, we performed a series of brainstorming meetings to elaborate on and discuss the main findings.

IV. Demographics

The participants have reported a minimum of 5 to a maximum of 33 years of *professional experience* in IT (15 years in average). At the time of migrating towards MSA, the participants were mainly involved as architects (8/18), CTOs (4/18) or developers (3/18), while 3 participants were involved with the specific role of DevOps engineer, industrial researcher and VP engineer. The *domains* of the systems being migrated vary significantly, as they include logistics, e-commerce, journalism, online gambling, finance, banking, healthcare and security.

The main customers of these systems were end-users (9/18) (i.e., software products such as Netflix or WhatsApp) and other companies (8/18) (i.e., the software is developed for and shipped to other companies, which will own it). In one case, the main customers were not known (1/18). Most participants classified their systems as Web-based (13/18). We consider a system as Web-based if it is developed with Web technologies, hosted on remote servers, served via standard protocols (e.g., HTTP), accessed via a unique URL.

With the definition of a monolith as *a software system whose modules cannot be executed independently* [6], almost all participants (17/18) classified their systems as a monolith. Interestingly, a subset of participants (5/17) specified that only the core of the pre-existing system was a monolith. The typical examples are systems that use functionalities implemented as independent services or provided externally by third parties. Five participants also stated that the pre-existing system could be (broadly) classified as service-oriented (SOA) (there defined as *any distributed system that is based on services and communicate over a network using standard communication protocols like SOAP and HTTP* [4]). Interestingly, all of them also answered that their systems were a monolith, either entirely (2/5) or the core part (3/5). We can conjecture that in these cases the monolith is a service of the SOA system that is the object of the migration.

When asked about the current status of the migration, the participants reported that in 6 cases the migration was still in its early stages, i.e. either only planned or just started (30% or less), while in 7 cases the migration was at 80% or 90% of its progress, and in 3 cases the migration was fully completed. The number of microservices deployed in the new system while the migration is still in progress varies from a minimum of 2 up to 250, with an average of 36. Differently, the number of microservices deployed upon completion of the migration in the new system varies from a minimum of 5 up to 250, with an average of 59. The expected time to fully migrate the system varies significantly among participants (from 9 to 60 months). On average, it is expected that the migration requires $\sim$28 months. The participants were also asked about the position of the management about the migration. In most of the cases, the management reacted positively to the migration, being easy to convince in (7/18) and supportive in (10/18). Maybe not too surprisingly, in some cases the management was not easy to convince (5/18) or not supportive (2/18). When asked about the alignment of the technical solution (IT) with business strategies, goals and needs (or 'Business-IT alignment'), more than half of the participants (11/18) have reported that the alignment was 80% or higher.

The participants also provided an estimation of both the number of teams and the average number of people per team, both at the beginning and upon completion of the migration. At the beginning of the migration the number of teams spanned from 1 to 20 (6.7 in average). Upon completion of the migration, the number of teams was estimated to be from 1 to 30 in the end (8 in average). The number of people per team before the migration spanned from 4 and 20, with an average of 8.3. Upon completion, the number of people per team ranged between 2 and 12, with an average of 6.5. We observe that, after the migration, the average number of

Authorized licensed use limited to: University of L'Aquila. Downloaded on September 08,2021 at 14:30:21 UTC from IEEE Xplore. Restrictions apply.

teams increased and the average number of people in each team decreased. This result is a confirmation of the fact that MSA encourages both small services and small teams (e.g., following the Amazon's notion of the Two Pizza Team [8]).

V. Migration Activities (RQ1)

A. Reverse Engineering

Table I shows the various forms of information about the pre-existing system which participants use when migrating towards MSA. Interestingly, participants relies on two important 'low-level' sources of information, respectively: the source code (13/18), and the test suites (11/18).

At an higher level of abstraction, the participants seem to prefer written documents in the form of: textual documents (11/18), architectural documents (e.g., architectural views) (11/18), data models or schema (10/18), and box and lines diagrams (9/18). Relevant knowledge about the system also resides inside the people within the organization (e.g., developers, architects, engineers) (9/18). Information is commonly shared by means of meetings (10/18), slides (8/18) and presentations (7/18). Less common among participants is the use of: UML diagrams (5/18), contracts with customers (3/18), architecture recovery tools for information extraction (1/18), and performance data (1/18). Although participants reported the source code as the main source of information on the pre-existing system, we have had a conflicting opinion about it, which we report below.

> "Of the old system we consider it as so bad that we do not look at the source code."

TABLE I
Information about the pre-existing system considered for the migration

Answers	Occurrences
Source code	13
Textual documents	11
Architectural documents	11
Tests	11
Meetings	10
Data models/schema	10
Box and lines diagrams	9
Unwritten knowledge among developers and engineers	9
Slides	8
Presentations	7
UML diagrams	5
Contracts with customers	3
Information extracted from architecture recovery tools	1
Other	1

During the migration, the information about the pre-existing system can be used for several purposes. As shown in Table II, participants have reported this information to be very important for understanding the pre-existing system (14/18), and for generating the knowledge for architecting the new system (14/18). In addition, the information about the pre-existing system can be relevant for: the definition of processes and APIs (12/18), the identification of dependencies in the pre-existing system (11/18), and the definition of tests for the new system (9/18). At some point during the migration, it seems rather frequent that this information is used also for deciding whether it is better (i.e., less costly in terms of budget or effort) to implement new functionalities in the pre-existing system or in the new system (6/18). Less common for participants is the need to use this information for: performing analysis (e.g., via metrics) on the pre-existing system (3/18), for improving its documentation (1/18), or for understanding what to keep or what to discard in the new system (1/18).

TABLE II
Purpose of the information about the pre-existing system

Answers	Occurrences
Understanding the pre-existing system	14
Architecting the new system	14
Defining processes and APIs in the new system	12
Finding dependencies in the pre-existing system	11
Creating tests for the new system	9
Trade-off analysis for deciding whether to implement new functionalities in the pre-existing system or in the new system	6
Perform analysis on the pre-existing system	3
Documenting the pre-existing system	1
Other	1

B. Architecture transformation

For investigating how practitioners design the new microservice architecture, we asked participants to identify what were the major activities involved in this process. As reported in Table III, respondents identified as the most relevant activities: domain decomposition (i.e., divide the domain into several subdomains) (11/18), service identification in the new architecture (11/18), application of domain-driven design (DDD) practices [7] (10/18), and system decomposition (i.e., cutting the pre-existing system into smaller components) (9/18). These four activities together indicate that it is of particular significance for practitioners to have a crystal clear understanding of the domain of the system in order to identify the bounded contexts properly. Presumably this allows practitioners to define sharp service boundaries in the new architecture and to start with the most suitable system decomposition technique for their systems. Immediately after these activities, participants identified as very important the need to get an immediate feeling (and directly experiment) with microservices: either by building proof-of-concept services to investigate the feasibility of the migration (8/18), or by implementing services as Minimum Viable Products (MVPs) (i.e., an artefact that may be incomplete in functionality or quality, but already showing the key characteristics for determining its customer value) [19] (7/18). Among the major activities performed when transforming the architecture, there is also the need to identify boundaries (8/18) and dependencies (7/18) in the pre-existing system.

Less common activities are related to the careful design of the business workflows processes (6/18), and the need to reduce risks in the network layer (3/18). An interesting quote about the risks in the network is the following.

> "Once you start going from the monolith to the microservice-based architecture your network component becomes a really big part of your system and as we all know the networking side of things it is notoriously difficult to debug."

Authorized licensed use limited to: University of L'Aquila. Downloaded on September 08,2021 at 14:30:21 UTC from IEEE Xplore. Restrictions apply.

TABLE III
ACTIVITIES PERFORMED WHEN DESIGNING THE NEW ARCHITECTURE

Answers	Occurrences
Domain decomposition	11
Identification of the services for the new system	11
Application of domain-driven design practices	10
System decomposition	9
Building proof-of-concept services to assess feasibility of the migration	8
Boundary identification in the pre-existing system	8
Implementing services as Minimum Viable Products	7
Dependencies identification in the pre-existing system	7
Careful design of the business workflows	6
Reduce risks in the network layer	3

When asked how the new architecture is documented, the participants have mainly reported the use of: architectural documents (14/18), textual documents (13/18), and box and lines diagrams (11/18). Presentations (9/18) and slides (6/18) are also commonly adopted for documenting the new design. In some cases, annotations and well structured source code are used for documenting the design (7/18). Differently, the adoption of semi-formal and formal modeling languages is less common among participants. Indeed, UML diagrams were reported only by 5 participants, while domain-specific language models only by 2 participants (cf. Table IV).

TABLE IV
HOW DID YOU DOCUMENT THE DESIGN OF THE NEW ARCHITECTURE?

Answers	Occurrences
Architectural documents	14
Textual documents	13
Box and lines diagrams	11
Presentations	9
Well written source code with annotations	7
Slides	6
UML diagrams	5
Notes	4
Domain-specific language models	2
Video	1
Other	1

When asked about the main driver of the migration, half of the participants have identified it in the functionalities (i.e., migrate iteratively according to new or prioritised functionalities). In the remaining cases, the main driver was very scattered. In two cases, the driver was the type of customer of the system (i.e., migrate according to customer's inputs and needs), while only in one case the driver of the migration was the management or the business (i.e., migrate according to the specification given by the management or business).
Participants have expressed as other drivers the following, each with one occurrence: (i) business-IT alignment (i.e., migrate trying to align as much as possible the services to the functional domain), (ii) customer processes separation (i.e. the system was split by analyzing each type of customer interaction), (iii) trade-off between costs and benefits of each migration of functionality (see quote below).

> "If we want a new feature or some performance gain or whatever incentive to change something that is already there, then we basically look at the cost that we sustained since we started building that part of the old system and we look at the cost that it would take to migrate to a separate system, and then we decide everyone to move forward and remove it from the monolith application, or if you want to keep it there and maybe add some layer or something like that. It depends on the benefits that we get and also on the expected investment that we want to make on that specific domain of the application."

Interestingly, in none of the cases the migration was driven by the data schema or data structure (i.e., migrate according to the necessary changes in the data structures) (cf. Table V).

TABLE V
DRIVER OF THE MIGRATION

Answers	Occurrences
The functionalities (i.e., migrate iteratively according to new or prioritised functionalities)	9
The type of customer of the system (i.e., migrate according to customer's inputs and needs)	2
Management/business (i.e., migrate according to the specification given by the management/business)	1
Other	4
Don't know	1
No information	1

As reported in Table VI, in 17 out of 18 cases the participants implemented new functionalities during the migration, as one of them nicely summarised:

> "It was like an upgrade of the system, not only a migration. Thinking in terms of workflows helped us comprehend better the business of our customers and so we ended up with an architecture that had better similarities to the actual business."

TABLE VI
WERE NEW FUNCTIONALITIES IMPLEMENTED DURING THE MIGRATION?

Answers	Occurrences
Yes	17
No	1

C. Forward Engineering

We have asked the participants to indicate how they actually started with the implementation of the migration. As reported in Table VII, participants either started the implementation by adding new functionalities as independent microservices (10/18) or by reimplementing existing functionalities in the pre-existing system as microservices (9/18). In 5 cases, the participant started directly by implementing the new system with a different architecture. Less common, but still interesting is the fact that some participants (4/18) started the migration by reimplementation existing functionalities as Minimum Viable Products.

To get more details on the initial phase of the migration, we asked participants to provide additional information on how the initial set of functionalities to migrate from the pre-existing system was identified (see Table VIII).

Authorized licensed use limited to: University of L'Aquila. Downloaded on September 08,2021 at 14:30:21 UTC from IEEE Xplore. Restrictions apply.

TABLE VII
HOW DID YOU START WHEN IMPLEMENTING THE MIGRATION

Answers	Occurrences
Implementing new functionalities as microservices	10
Reimplementing existing functionalities as microservices	9
Implementing the new system	5
Reimplementing existing functionalities as Minimum Viable Products	4
Other	1
No information	1

In 6 cases, participants have reported that they identified the functionalities among the ones with less dependencies in the pre-existing system. Presumably this choice allows to extract the functionalities with little impact on the architecture of the pre-existing system. In 2 cases, the functionalities were identified among the less used by the users (see quote below). In the remaining cases, the initial set of functionalities were identified as follows: (i) by the MVP requirements, (ii) among the most important to the customers, (iii) extraction of integration code based on existing CQRS/event-based architecture, and (iv) among functionalities affected by problems that were difficult to fix on the monolith. Notably, none of the participants have selected the functionalities to migrate among the less used in the system.

> ”Once the architecture was defined, we started to test how the [pre-existing] system would react. We started to migrate the parts which had minor impact on the users, mainly for testing purposes and to get more confidence. If things went wrong, then this would not have a big impact on the system, otherwise, if everything went well, you could already test if and how much the improvement was important. You could immediately notice it.”

TABLE VIII
HOW DID YOU IDENTIFY THE FIRST FUNCTIONALITIES TO MIGRATE?

Answers	Occurrences
Less dependencies	6
Less used by the users	2
Other	4
Don’t know	2
No information	4

Practitioners were asked what method they used for adopting the new systems in production (see Table IX). Mainly, the new system was deployed with a phased adoption (i.e., the new system is created in a series of pre-determined steps and over a period of time) (14/18). The well-known strangler pattern[2] belongs to the phased adoption. In 3 cases, the participants reported a parallel adoption (i.e., both the complete old system and the new system run simultaneously for some time after which, if the criteria for the new system are met, the old system is disabled). Interestingly, in one case a big bang adoption (i.e., the old system moves to the new system in a single major event) was used.

[2]https://www.martinfowler.com/bliki/StranglerApplication.html

TABLE IX
HOW DID YOU ADOPT THE NEW SYSTEM?

Answers	Occurrences
Phased adoption	14
Parallel adoption	3
Big bang adoption	1

Table X shows how participants have handled the pre-existing data during the transition to the new system. In the majority of cases (11/18), the data was kept ’as is’, without modifying its structure. Differently, in 2 cases the data was migrated to the new system by implementing data migration procedures and making the new services responsible for handling both old and new data. In 5 cases, participants have reported that either there was no need to migrate the data (3/5) or the services were mostly dataless (2/5).

TABLE X
HOW DID YOU CONSIDER PRE-EXISTING DATA IN THE NEW SYSTEM?

Answers	Occurrences
The data was kept ’as is’ in the pre-existing system	11
The data was migrated to the new system by implementing data migration procedures. New services handle both old and new data	2
Other	5

VI. CHALLENGES (RQ2)

In this section we analyse the challenges faced by participants in the three subprocesses of the horseshoe model.

A. Reverse Engineering

The main challenges faced by participants in the pre-existing system are reported in Table XI. Almost all participants (15/18) have acknowledged the long time to release new features as the main challenge faced in the pre-existing system. From an architectural perspective, we can see how participants find challenging the high degree of coupling among modules (13/18) in the pre-existing system.

> [The main challenge was] ”the high degree of coupling: creating new features was longer and longer, and much harder.”

Moreover, one participant reported the coupling even in the process.

> ”The problem was that the coupling was in the process: when we built the whole thing it was built in a huge client-server process. It was coupling together all modules.”

Significant on the architecture level is the fact that pre-existing systems are hard to maintain (13/18) and sometimes hard to test (7/18). On a more technical perspective (i.e., the developers’ perspective), a significant challenge is related to side effects.

> [One of the events behind the decision of changing the architecture] ”I was the architect at the time. Once a developer came in and told me, ’would you mind taking a look at my code change, because last time I changed something in this module, it

Authorized licensed use limited to: University of L'Aquila. Downloaded on September 08,2021 at 14:30:21 UTC from IEEE Xplore. Restrictions apply.

had a side effect in this one which crashed a third one'. The three modules were nothing related to each other."

The participants also faced challenges concerning the low developers' productivity and developers interfering with each other (7/18). Some other challenges are related to the lack of a proper documentation of the system (6/18), knowledge degradation (5/18), and unknown boundaries of functionalities/modules (5/18). In 4 cases, the participants identified a challenge in convincing the business/management that migration was a necessary step to perform.

TABLE XI
MAIN CHALLENGES - REVERSE ENGINEERING

Challenges	Occurrences
Long time to release new features	15
High coupling	13
Hard to maintain	13
Side effects	10
Low productivity of developers	9
Hard to test	7
Developers interfering with each other	7
Lack of documentation	6
Knowledge degradation	5
Unknown boundaries of functionalities or modules	5
Poor confidence on the source code	4
Convincing the business/management that migration was necessary	4
Making information explicit	4
Issue with the programming language	2
Other	4
Don't know	1

B. Architecture Transformation

Table XII reports the challenges faced by participants when transforming the pre-existing architecture in the new architecture. The most relevant challenge is the high level of coupling among parts of the pre-existing system (9/18). Understandably, when transforming the architecture towards MSA, the more the modules are coupled with each other, the more difficult it is to extract functionalities from the pre-existing system. Other challenges include: the identification of service boundaries (7/18), decomposition of the pre-existing system (6/18), and the need to reduce the coupling among services in the new architecture (6/18). Participants have also reported as challenging the configuration/setup of the automation support for testing (e.g., in continuous integration and continuous delivery pipelines) (6/18). Less common challenges are related to the Business-IT alignment (5/18), domain decomposition (4/18), and the difficulties in bringing domain experts into the process of designing the new system (2/18). Among the other, some participants reported as challenging the following: (i) difficulties in taming the project, the people and the management, and also find agreement on the architecture as too many people were involved, (ii) implementing new features in the old/new system while migrating, (iii) the need to change the MVP according to business goals modifications, keeping up with changes still being made to the existing system was really hard, making parts of the new system still work with the old system was technically challenging, and the communication between stakeholder and developer teams was poor.

TABLE XII
MAIN CHALLENGES - ARCHITECTURE TRANSFORMATION

Challenges	Occurrences
High coupling among parts of the pre-existing system	9
Identification of the boundaries of each service	7
Decomposition of the pre-existing system	6
Automation support for testing	6
Reduce coupling among services in the new architecture	6
Finding the best Business-IT alignment	5
Decomposition of the domain	4
Finding the proper service granularity	4
Lack of proper documentation of the system	3
Bring domain experts into the process of designing the new system	3
Undocumented/uncommented code	2
Other	6

C. Forward Engineering

Table XIII reports the challenges faced when implementing the new system. Half of the participants have acknowledged the main challenge in the setup of the initial infrastructure for microservices to properly work. Immediately after, the different mindset for developers is among the most relevant challenges (see quote below).

> "Most of the developers were quite used to get everything in one database, so if they wanted some information, they would just pick it up from the right table. And now you have got to make them understand that they have got to do an HTTP call to get this data. So they have got to think in a completed different way because an HTTP call is expensive. It is going to add some matters of authentication, identification, when you call one microservice to the other it could be through a dedicated account, while before you would just make the SQL."

TABLE XIII
MAIN CHALLENGES - FORWARD ENGINEERING

Challenges	Occurrences
Setting up the initial infrastructure for microservices to work	9
Different thinking for developers	7
Distributed monitoring	6
Knowledge sharing, effective communication	6
Distributed logging	5
Distributed debugging	5
Create uniformity across services	5
Testing the new system	4
Get the initial team to work together	4
Using standards and norms	2
Get the initial prototype working	1
Other	6

Knowledge sharing was also perceived as challenging when implementing the new architecture (6/18). Distributed aspects related to monitoring (6/18), logging and debugging (5/18) were also reported as challenging. An interesting challenge is related to achieving uniformity across services.

> "I think actually that the biggest problem that we try to tackle is uniformity across all of our services, so once you start working with 30 services you want a mix between creating freedom for your engineers to

Authorized licensed use limited to: University of L'Aquila. Downloaded on September 08,2021 at 14:30:21 UTC from IEEE Xplore. Restrictions apply.

> experiment and try out new systems and new frameworks etc., but at the same time you want uniformity, so that a single engineer does not have to learn 20 different languages or 10 different frameworks to work with all the systems that we have within the company. So uniformity is the biggest challenge that we have."

Some additional challenges involve: testing the new system (4/18) and getting the first team to work together (4/18). Some participants have reported more specific challenges: (i) we were afraid of the impact on the system of the new services being deployed as microservices (it was partially unknown), (ii) work together with the business, and get business availability to do the testing in the right way, (iii) needed infrastructure for container management, service discovery and registration, (iv) even though the development of the initial system went really well, the transition plan became the actual problem of the migration.

VII. Discussion

A. Reflections on the obtained findings

A first reflection is on the *migration process* itself. As reported by two participants, in some cases the migration towards MSA is organized in small increments, rather than a big overall migration project. In those cases, the migration is implemented as an iterative and incremental process, as also confirmed by the high number of occurrences reported for the phased adoption (cf. Table IX). A similar observation has been reported for migrations towards SOA in the industrial survey of Razavian and Lago [24]. Moreover, it must be noted that in some cases the migration has a starting point, but not necessarily a defined-upfront end-point. As an example of this recurrent phenomenon, see the following quote from one of the participants.

> "The most important thing is that there is no reengineering project or something that is as a single goal of rebuilding the system, we are doing that as part of our daily work."

In migrating towards MSA, semi-formal models (i.e., UML specifications) and domain-specific models are not used much, neither for designing the architecture nor for describing the migration. A similar trend about the little application of models was observed in the literature about MSA [5], with some exceptions [11], [12], [28], [29].

Agility seems to be a very relevant aspect when migrating towards MSA. Almost all participants added new functionalities during the migration, which means that the pre-existing system was impeding the addition of new functionalities. This is also confirmed by the challenges faced with the pre-existing system (cf. Table XI). If we consider the information we gathered on the migration implementation, we can notice that participants start implementing new functionalities or reimplementing existing functionalities as microservices, but they also reported that one of the most challenging task is the setup of the initial infrastructure for running microservices (cf. Table XIII). We can conjecture that there is room for improvement here, e.g., by applying Domain-Driven Design practices. Moreover, there could be room for improvement also in the techniques for the identification of the initial microservices to be migrated (cf. Table VIII).

Surprisingly, more than half of the participants have reported that existing data during the migration is not being migrated (cf. Table X). This does not align well neither with the "*hide internal implementation detail*" principle of microservices [20], nor with the typical MSA characteristic of *decentralized data management* [8]. Not migrating data may possibly hinder the ability of evolving services independently, but also the possibility to scale up both services and their data. However, it seems reasonable to think that if companies do not have stringent needs for scalability, they do not put effort in the migration of data.

Regarding architecture transformation, the main challenges are (i) the high level of coupling, (ii) the difficulties in identifying the boundaries of services, and (iii) system decomposition. There is room for improvement in this area, for example, by providing techniques for the identification of services to be migrated at the architectural level, possibly with the support of architecture recovery tools.

Finally, from the data we can observe that participants are reaching a good level of Business-IT alignment. As also fundamental for SOA applications, this is promising and an indication of the fact that practitioners are building their microservice-based systems by following the "*model around business concepts*" of microservices [20].

B. Action points

Even though the sample of our survey is small, and taking into account the threats to validity of our study (see Section VIII), we have elaborated a few action points for practitioners and researchers that are worth considering. For practitioners, we identified three main action points. First, **share** your success stories: build and share reusable technical competence/knowledge, (i) to kickstart a MSA, and (ii) to reuse solutions. Second, **check** business-IT alignment: that's a key concern during migration. Third, **monitor** the development effort and **migrate** when it grows too much: we observed a relatively high correlation between migration to microservices and increasingly prohibitive effort in implementing new functionalities in the monolith.
For researchers, we have identified a main action point, which is: **address** the (apparently open problem) of how to migrate pre-existing data (e.g., legacy databases) to microservices.

VIII. Threats to Validity

We designed and conducted this study so to avoid biases as much as possible. Specifically, we have been very careful while characterizing the target population, in the design and wording of the questionnaire, and during the data analysis and synthesis phases. In the following the main threats to validity and our corresponding mitigation strategies are reported according to the Cook and Campbell classification [3].
Conclusion validity. It concerns the relationship between the extracted data and the obtained results [27]. In this context, one

Authorized licensed use limited to: University of L'Aquila. Downloaded on September 08,2021 at 14:30:21 UTC from IEEE Xplore. Restrictions apply.

of the most important threats to validity relates to the fact that other researchers may identify questions and possible closed-ended answers different from the ones in our questionnaire. We mitigated this potential threat by (i) letting the closed-ended responses of the questionnaire emerge from the data obtained from the interviews, (ii) piloting the online questionnaire several times with the help of independent researchers, and (iii) having the data extraction process conducted and checked by two researchers. Moreover, our study involves 18 participants across the world and at different professional stages. We are aware that the sample size of our study is limited with respect to the potential large set of MSA practitioners and that this prevents us to perform a statistical analysis of the obtained data [14]. This is a consequence of the specificity of the targeted population (i.e., practitioners who have been directly and recently involved in a migration towards microservices). We mitigated this potential threat by do not performing any statistical correlation analysis (which may have inevitably led to a low statistical power).

Internal validity. It refers to the causality relationship between treatment and outcome [27]. Firstly, we employed basic descriptive statistics during the data analysis phase, so the threats with respect to the correctness of the performed analysis were minimal. During the data analysis we also cross-analyzed the answers of different questions of each participant in order to make a sanity test of the extracted data. Secondly, we did not have the possibility of directly interacting with participants filling the online questionnaire, potentially risking them to do not fully understand the questions and their possible closed-ended responses. Also, in those cases we could not ask follow-up questions to the participants. We mitigated this potential bias by (i) providing a set of definitions at the beginning of the questionnaire that are used consistently in the survey to help them to answer the questions, (ii) asking a set of open-ended questions in which participants may freely give their comments on specific aspects of their migration to microservices, and by contacting participants by email with the link to the online questionnaire; this allowed participants to fill in the questionnaire as soon as they had time to do it, and to reason about their answers as long as they wished.

Construct validity. It deals with the relationship between theory and observation [27]. One of the most recurrent threats in questionnaire-based surveys regards the phrasing adopted to define statements, questions, and responses for closed-ended questions. Similarly to what we did for mitigating threats to conclusion validity, we let the closed-ended responses emerge from the interviews and piloted the online questionnaire. Moreover, during the design of the questionnaire, we also piloted the questionnaire internally several times and refined the language in each of its parts extensively. Finally, since the questionnaire contains several closed-ended questions, potentially we could have risked that for some questions participants did not find any suitable response in the set of available ones. We mitigated this risk by (i) defining the set of responses of each question by systematically analysing the interview notes, and (ii) by always including an ”Other” alternative for each question; i.e., an open textual field which allows participants to freely add their own answers. As a confirmation of the comprehensiveness of our questionnaire, a relatively low number of participants used such a free textual field.

External validity. It concerns the generalizability of obtained results [27]. As described in Section III, in this study we applied a combination of convenience and snowball sampling. Convenience and snowball sampling helped us in selecting study participants in a precise and accurate manner. However, we are aware that this choice may have had a negative impact on the size of the set of subjects of our study. We mitigated this potential threat to validity by ensuring that they are an heterogenous sample in terms of professional experience, size and domain of their company, migration phases in which they are involved, etc. (refer to Section IV for the details about the demographics of the participants). This indicates a good spread of participants' profiles and increases the confidence that the results of this study do not suffer from severe biases with respect to the sampling method we applied, despite the small sample size.

IX. Related Work

Taibi et al. [25] have performed a questionnaire survey on migration towards MSA, filled by 21 practitioners. Although their approach is similar to ours, their focus differ deeply. In our work we perform an in-depth investigation of migration activities and challenges, considering them in great detail in each phase related to: the recovery of the pre-existing system architecture, architecture transformation, and the development of the new system architecture. Taibi et al., in turn, just skimmed on these aspects, and instead analyzed in greater depth other aspects of the migration, like the motivations behind the migration to MSA, and the benefits they provide. Another significant difference between the two works is the fact that Taibi et al. focus only on migration projects from monolithic applications to MSA, while we consider also the cases when the migration is performed starting from SOA.

To the best of our knowledge, no other investigations have been performed on migration practices to MSA. A similar study on SOA migration approaches has been performed by Razavian and Lago [24]. This work differs from ours for the specific target of the migration, as they focus on SOA instead of MSA. A number of secondary studies have been published on the topic of microservices. In their work, Pahl and Jamshidi [22] identify migration towards microservices among the major research trends. Similarly, the results of our mapping study on architecting microservices [5] confirm that migration towards MSA is one of the most relevant research perspectives. Differently, two secondary studies of Alshuqayran et al. [1] and Vural et al. [26] do not directly address the migration towards MSA in their works. All the above-mentioned secondary studies differ from our work in two ways: we directly involve industrial practitioners; and we specifically focus on the migration activities and challenges for the adoption of MSA.

Balalaie et al. [2] generalize their experience by defining and describing an initial set of microservices migration patterns.

Authorized licensed use limited to: University of L'Aquila. Downloaded on September 08,2021 at 14:30:21 UTC from IEEE Xplore. Restrictions apply.

These patterns can be used as a reference during the planning phase of a migration towards microservices.

More in general, a number of primary studies, e.g., [9], [10], [16], [17] have addressed somehow the topic of migration towards MSA (e.g., techniques, approaches, analyses, experience reports); however, they differ from our work as we aim at identifying more generalizable and common insights from industry while the focus of the primary studies is typically on specific case study or domains. A complete reference of primary studies is out of scope of this research.

X. Conclusions and Future Work

This paper presents an empirical study of how practitioners deal with the migration of software systems towards microservice architectures. We first performed exploratory interviews with 5 practitioners with the aims of (i) becoming familiar with the practitioners' perspectives on the topic and (ii) guiding the design of an online questionnaire. The questionnaire has been proposed to a number of practitioners among our collaboration network and eventually 13 of them successfully completed it. Both the questionnaires and the interviews have been structured around the horseshoe migration model proposed by Kazman et al. [13].

The data extracted from the interviews and questionnaires shed light on the various activities and challenges faced by practitioners when migrating to MSA, which can benefit both (i) researchers who are interested in better shaping their methods and techniques according to the actual state of the practice about the migration to MSA, and (ii) practitioners who are planning to migrate to MSA and need to learn from how such a potentially disruptive activity is performed in other industrial contexts.

As future work, we are planning to design an architecture-centric method for supporting the three migration phases of the horseshoe model in the context of MSA. Such a method will be based on the model-driven engineering paradigm, where architecture models will be represented by means of a dedicated domain-specific language for MSA. Having a domain-specific-language-based representation of an MSA will empower architects to (semi-)automatically extract, transform, and re-engineer existing systems, while preserving key properties (e.g., models consistency, correctness) across each step of the migration. Furthermore, we plan to expand the sample size with the goal of enhancing our analysis and improve its representativeness for migration.

References

[1] N. Alshuqayran, N. Ali, and R. Evans. A Systematic Mapping Study in Microservice Architecture. In *Proc. of the 9th International Conference on Service-Oriented Computing and Applications*. IEEE, IEEE, 2016.

[2] A. Balalaie, A. Heydarnoori, and P. Jamshidi. Microservices migration patterns. Technical report, Tech. Rep. TR-SUTCE-ASE-2015-01, Automated Software Engineering Group, Sharif University of Technology, Tehran, Iran, 2015.

[3] T. D. Cook, D. T. Campbell, and A. Day. *Quasi-experimentation: Design & analysis issues for field settings*, volume 351. Houghton Mifflin Boston, 1979.

[4] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. Unraveling the web services web: an introduction to soap, wsdl, and uddi. *IEEE Internet computing*, 6(2):86–93, 2002.

[5] P. Di Francesco, I. Malavolta, and P. Lago. Research on architecting microservices: Trends, focus, and potential for industrial adoption. In *IEEE International Conference on Software Architecture (ICSA)*, pages 21–30. IEEE, 2017.

[6] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina. Microservices: yesterday, today, and tomorrow. *arXiv preprint arXiv:1606.04036*, 2016.

[7] E. Evans and R. Szpoton. *Domain-driven design*. Helion, 2015.

[8] M. Fowler and J. Lewis. Microservices a definition of this new architectural term. *URL: http://martinfowler.com/articles/microservices.html*, Last accessed: Jan 2018.

[9] A. Gopu, S. Hayashi, M. D. Young, R. Kotulla, R. Henschel, and D. Harbeck. Trident: Scalable compute archives-workflows, visualization, and analysis. 2016.

[10] J.-P. Gouigoux and D. Tamzalit. From monolith to microservices: Lessons learned on an industrial migration to a web oriented architecture. In *Software Architecture Workshops (ICSAW), 2017 IEEE International Conference on*, pages 62–65. IEEE, 2017.

[11] G. Granchelli, M. Cardarelli, P. Di Francesco, I. Malavolta, L. Iovino, and A. Di Salle. Towards recovering the software architecture of microservice-based systems. In *Software Architecture Workshops (ICSAW), 2017 IEEE International Conference on*, pages 46–53. IEEE, 2017.

[12] G. Granchelli, M. Cardarelli, P. D. Francesco, I. Malavolta, L. Iovino, and A. D. Salle. Microart: A software architecture recovery tool for maintaining microservice-based systems. In *Proceedings of the 14th International Conference on Software Architecture (ICSA)*, pages 298–302. IEEE, 2017.

[13] R. Kazman, S. G. Woods, and S. J. Carrière. Requirements for integrating software architecture and reengineering models: Corum ii. In *Reverse Engineering, 1998. Proceedings. Fifth Working Conference on*, pages 154–163. IEEE, 1998.

[14] L. Kish. *Survey sampling*. Wiley Classics Library. Wiley, 1995.

[15] B. Kitchenham and S. L. Pfleeger. Principles of survey research: part 5: populations and samples. *ACM SIGSOFT Software Engineering Notes*, 27(5):17–20, 2002.

[16] H. Knoche. Sustaining runtime performance while incrementally modernizing transactional monolithic software towards microservices. In *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering*, pages 121–124. ACM, 2016.

[17] D. S. Linthicum. Practical use of microservices in moving workloads to the cloud. *IEEE Cloud Computing*, 3(5):6–9, 2016.

[18] M. Miles and A. Huberman. *Qualitative Data Analysis: An Expanded Sourcebook*. Sage, Thousand Oaks, 2 edition, 1994.

[19] J. Münch, F. Fagerholm, P. Johnson, J. Pirttilahti, J. Torkkel, and J. Jäarvinen. Creating minimum viable products in industry-academia collaborations. In *Lean Enterprise Software and Systems*, pages 137–151. Springer, 2013.

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

[21] A. N. Oppenheim. *Questionnaire design, interviewing and attitude measurement*. Bloomsbury Publishing, 2000.

[22] C. Pahl and P. Jamshidi. Microservices: A Systematic Mapping Study. In *Proceedings of the 6th International Conference on Cloud Computing and Services Science, Rome, Italy*, pages 137–146, 2016.

[23] M. Razavian and P. Lago. Understanding SOA migration using a conceptual framework. *Journal of Systems Integration*, 1(3), 2010.

[24] M. Razavian and P. Lago. A survey of SOA migration in industry. In *ICSOC*, pages 618–626. Springer, 2011.

[25] D. Taibi, V. Lenarduzzi, and C. Pahl. Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation. *IEEE Cloud Computing*, 4(5):22–32, 2017.

[26] H. Vural, M. Koyuncu, and S. Guney. A systematic literature review on microservices. In *International Conference on Computational Science and Its Applications*, pages 203–217. Springer, 2017.

[27] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering*. Computer Science. Springer, 2012.

[28] E. B. H. Yahia, L. Réveillère, Y.-D. Bromberg, R. Chevalier, and A. Cadot. Medley: An event-driven lightweight platform for service composition. In *International Conference on Web Engineering*, pages 3–20. Springer, 2016.

[29] M. Zúñiga-Prieto, E. Insfran, S. Abrahao, and C. Cano-Genoves. Incremental integration of microservices in cloud applications. 2016.

Authorized licensed use limited to: University of L'Aquila. Downloaded on September 08,2021 at 14:30:21 UTC from IEEE Xplore. Restrictions apply.