Towards an Architecture-Centric Methodology for Migrating to Microservices

Jonas Fritzsch[1](✉), Justus Bogner[3], Markus Haug[1], Stefan Wagner[1], and Alfred Zimmermann[2]

[1] University of Stuttgart, Stuttgart, Germany
{jonas.fritzsch,markus.haug, stefan.wagner}@iste.uni-stuttgart.de

[2] University of Applied Sciences Reutlingen, Reutlingen, Germany
alfred.zimmermann@reutlingen-university.de

[3] Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
j.bogner@vu.nl

Abstract. The euphoria around microservices has decreased over the years, but the trend of modernizing legacy systems to this novel architectural style is unbroken to date. A variety of approaches have been proposed in academia and industry, aiming to structure and automate the often long-lasting and cost-intensive migration journey. However, our research shows that there is still a need for more systematic guidance. While grey literature is dominant for knowledge exchange among practitioners, academia has contributed a significant body of knowledge as well, catching up on its initial neglect. A vast number of studies on the topic yielded novel techniques, often backed by industry evaluations. However, practitioners hardly leverage these resources. In this paper, we report on our efforts to design an architecture-centric methodology for migrating to microservices. As its main contribution, a framework provides guidance for architects during the three phases of a migration. We refer to methods, techniques, and approaches based on a variety of scientific studies that have not been made available in a similarly comprehensible manner before. Through an accompanying tool to be developed, architects will be in a position to systematically plan their migration, make better informed decisions, and use the most appropriate techniques and tools to transition their systems to microservices.

Keywords: microservices · refactoring · software architecture

1 The Challenge of Moving to Microservices

In times of cloud-based software solutions, the microservices architectural style has become the de facto standard for large-scale and cloud-native commercial applications [15]. Technological advancements like containerization and automation have paved the way for efficiently operating almost any number of independent functional units. However, existing legacy systems are often designed as monoliths and can therefore barely benefit from advantages such as improved scalability, maintainability, and agility through independent deployment units [11]. Hence, many companies try to migrate their systems towards microservices. While a rewrite of the entire application is expensive and often infeasible, architects are looking for less resource-intensive approaches to modernize a system. Architectural refactorings that are (partly) automated may reduce effort and risk of a migration tremendously. Unfortunately, there is no general approach that fits for arbitrary systems [10]. A thorough analysis is required to choose the appropriate strategy and refactoring technique.

Early adopters of microservices had to deal with manifold challenges, such as a lack of technical guidance and best practices, immature tooling, or organizational aspects. While pioneers like Amazon, Netflix, Spotify, and even the German retailer Otto published on their journeys of microservices adoption, such exemplary cases do not necessarily qualify as a blueprint. The average enterprise system has often more sophisticated tasks to fulfill than the well-studied retail domain, and moreover, IT budgets are often tight. As well, strict compliance and high-quality requirements do not leave much room for experimentation and failed investments. Hence, architects often struggle to find suitable guidance on planning and conducting such an architecture change systematically. A migration and in particular the decomposition of industry-scale legacy systems is a seen as major challenge in this regard [9]. While practitioners often achieve a reasonable solution with extensive manual efforts, the targeted and quality-controlled planning of a migration as well as a semi-automated decomposition continue to be problematic. In the following two subsections, we briefly summarize our view on the typical progress of a system migration to microservices and describe the gap between academia and industry.

1.1 Three Phases of a Migration

Based on our groundwork and existing research [5,14,16], we identify three main phases of a migration: *system comprehension*, *planning*, and *implementation*. The initiation of a migration is commonly started with comprehending the existing system: After the definition of strategic goals, quality requirements are determined by all stakeholders of the system. They serve as a measure for assessing the legacy system and potential alternatives. Hence, the outcome of this first phase should be a grounded decision for or against a modernization, based on specific quality attributes and metrics. While monolith and microservices are not the only possible architectural styles, we exclusively focus on the contraposition of these two patterns.

Given that the comprehension phase resulted in favor of a migration, the planning phase aims at defining an adequate strategy. One of the two major tasks in this phase is the definition of a development process based on different types of software modernization [3]. Distinguishing greenfield and brownfield developments[1], we further split up the second in a re-build or re-factor development type. Further distinctions can be made that affect the time frame and consumption of resources. While a *big bang*[2] migration aims to minimize the duration, a continuous evolution strategy tries to minimize the needed resources. The second decision in the planning phase concerns the choice of a service identification approach. It yields a suitable service cut (decomposition of the existing system) and thereby determines the granularity of resulting services. Our previous research has shown that deciding on the decomposition is often a manual task [9] guided by methods like Domain-Driven Design (DDD). However, a variety of existing artifacts from the legacy system can be beneficial for automating this task to some extent, e.g., code bases, databases, version control system data, runtime logs and traces, and various other design documents or models.

After completion of the initial two phases, the actual implementation starts. The elaborated strategy implies boundaries for duration, required resources, as well as needed organizational changes. The latter are in particular relevant as a consequence of altering a system's architecture [7]. Microservices candidates and target architecture are defined, based on the approach and techniques chosen in the planning phase. The now following implementation of services commonly iterates through several cycles. In each cycle, one or more microservices are implemented, accompanied by a quality assessment of the emerging target system. That way, inadequacies of the architecture definition or even an unsuitable decomposition can be corrected early with reasonable effort. The organizational changes, infrastructure build-up, and establishment of DevOps processes go hand in hand with the migration progress.

1.2 The Academia-Industry Gap

A rapidly growing number of scientific publications deal with the topic of microservices migrations, as the meta studies by Schroer et al. [13] and Ponce et al. [12] show. Existing research covers a variety of topics, starting from decision-making over process strategies [3] to quality assurance [6] and organizational aspects [9]. The challenging question of service identification techniques in general [1] and for microservices specifically is targeted by several dozen studies [10,12]. However, our empirical research has shown that this extensive body of scientific literature is mostly unknown to practitioners and therefore rarely leveraged [9]. We found that even specialized consultancy companies do rarely consider such knowledge. There may be several aspects to this barrier, e.g., reservations regarding scientific databases, access limitations, or concerns regarding the practical applicability and relevance of scientific research. Hence, a key motivation of our work is in filtering, pre-processing, and presenting the relevant works to practitioners based on their specific systems and migration scenarios.

[1] Greenfield development refers to creating a system for a new environment from a clean slate, no legacy code is required. Brownfield developments require the presence of an existing system that gets improved: data, processes and settings are retained.

[2] Migration within a limited time window and instant switch from old to new system.

2 Research Design

Figure 1 illustrates the overall research method. Our research objective is framed by the following two questions:

RQ1: How can a process framework represent a holistic view on microservice migration activities with a focus on architectural refactoring techniques?
RQ2: How can tool support based on such a framework provide guidance for architects in a specific migration scenario?

As a foundation, we analyzed existing literature on the microservice migration process. In an interview study among 16 practitioners from 10 companies, we analyzed 14 systems from various domains regarding intentions, practices, and challenges [9]. In addition, we contributed an early meta study classifying architectural refactoring approaches for migrations to microservices [10]. Our resulting methodology serves as a basis for longitudinal case studies that are currently conducted in cooperation with industry (including DATEV eG and Siemens AG). In an iterative process, the framework and accompanying tool support will be evaluated and refined. As a final step, we plan a large-scale survey among practitioners to assess the accompanying tools' applicability in certain contexts, its usefulness, and usability.

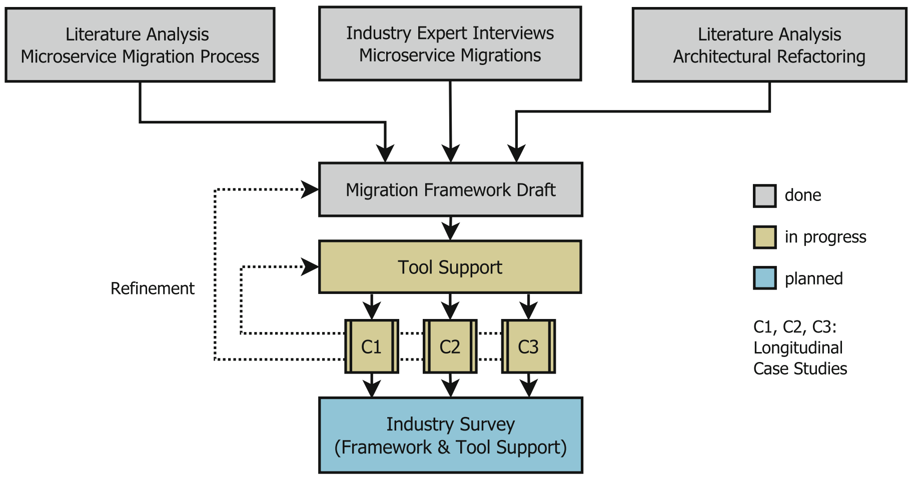
Literature Analysis Microservice Migration Process
Industry Expert Interviews Microservice Migrations
Literature Analysis Architectural Refactoring
Migration Framework Draft
Tool Support
Refinement
C1
C2
C3
Industry Survey (Framework & Tool Support)
done
in progress
planned
C1, C2, C3: Longitudinal Case Studies

Fig. 1. Research Method

3 Related Work

According to the outlined research method, we split up the discussion of related studies into three clusters: 1) migration process, 2) architectural refactoring, and 3) associated aspects like quality assurance and re-organization.

1) Migration Process. In their survey among 18 practitioners, Di Francesco et al. collected the various activities carried out in a migration to microservices [8]. The work provides an empirically collected, bottom-up classification of common activities. Taibi et al. followed a similar survey-based approach when querying 21 practitioners [14]. They reconstructed a migration process framework that reflects the interviewees' procedures and best practices. In addition to Di Francesco et al., they also distinguish between re-development and continuous evolution strategies, applying the popular Strangler pattern. Wolfart et al. approach the migration topic more holistically in their migration roadmap [16]. They analyzed six primary studies to come up with a unified process. To this end, they conducted a more comprehensive systematic mapping of 62 primary studies dealing with the modernization of legacy systems to microservices. Their resulting framework depicts eight activities grouped into four phases, namely Initiation, Planning, Execution, and Monitoring. In the same way, we can regard the roadmap by Bozan et al. [5] on incrementally transitioning to a microservices architectures, which was distilled from interviews with 31 software experts. In contrast to the above-mentioned studies, the authors here also reflect on the organizational and business-related impacts.

2) Architectural Refactoring. The secondary studies by Abdellatif et al. [1], Bajaj et al. [3], Schroer et al. [13], Ponce et al. [12], and Fritzsch et al. [10] provide a holistic overview of this major technical challenge in a migration. Our earlier study [10] attempted a classification of approaches based on their underlying techniques. Abdellatif et al. [1] developed a more elaborate taxonomy that provides a solid foundation for use in our framework. The majority of studies can be ascribed to at least one of the three basic categories of techniques: model-driven, static analysis, and dynamic analysis [12]. In addition, organizational structures or metadata like version control history [10] can also provide valuable input for the architectural refactoring.

3) Associated Aspects. As initially set strategic goals and subsequently identified quality attributes largely steer the architecture transformation, quality assurance needs a strong focus, as outlined by Shahin et al. [13]. This aspect is reflected in some of the above suggested frameworks during the initial phase (requirements and strategic goals) or in the form of verification and validation activities. As a basis for assessing the relevance of different quality attributes for microservices in general [4] and in the context of a migration [9], we build upon our earlier empirical research. It also revealed that organizational changes and social aspects such as a mindset change can have considerable impact on the migration process.

Significant advances have been made in detailing the process of a migration, as well as in elaborating techniques for decomposing monolithic systems. However, there is no holistic methodology available that combines both aspects. In addition, our research revealed the lack of a vehicle for knowledge transfer into practice. We aim to address this issue by suggesting a methodology that presents a holistic view on microservice migration activities, with a focus on architectural refactoring. It is enriched with a systematic quality assessment and aims for a high degree of automation. Furthermore, we seek to develop a supplemental tool that guides architects, thereby allowing them to efficiently leverage the comprehensive body of scientific knowledge.

4 Proposed Migration Framework

To address RQ1, we propose an architecture-centric framework for migrating to microservices as depicted in Fig. 2. It incorporates ideas and groundwork of existing research, especially the works by Wolfart et al. [16]. Aspects of the works by Taibi et al. [14] and Bozan et al. [5] have influenced the design as well. According to the discussion in Sect. 1.1, we split a migration into three phases that are detailed below. Each activities' result artifacts are shown in a flowchart on the right side in Fig. 2. The reflected process may be applied separately for single subsystems as required.

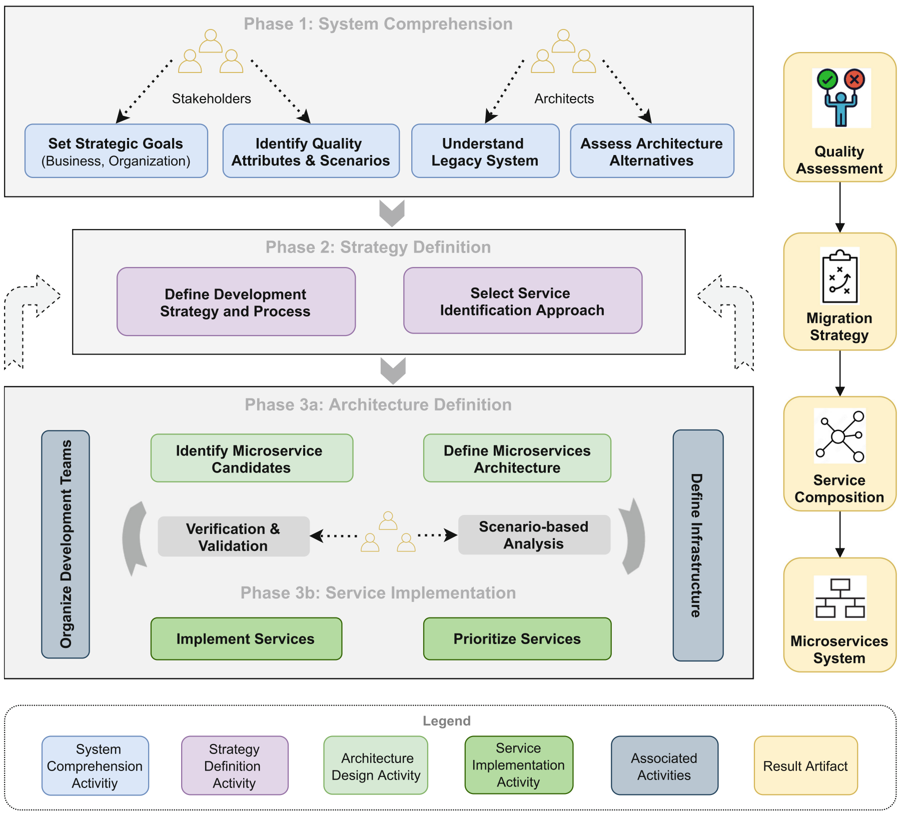
Phase 1: System Comprehension
Stakeholders
Architects
Set Strategic Goals (Business, Organization)
Identify Quality Attributes & Scenarios
Understand Legacy System
Assess Architecture Alternatives
Phase 2: Strategy Definition
Define Development Strategy and Process
Select Service Identification Approach
Phase 3a: Architecture Definition
Organize Development Teams
Identify Microservice Candidates
Define Microservices Architecture
Define Infrastructure
Verification & Validation
Scenario-based Analysis
Phase 3b: Service Implementation
Implement Services
Prioritize Services
Quality Assessment
Migration Strategy
Service Composition
Microservices System
Legend
System Comprehension Activitiy
Strategy Definition Activity
Architecture Design Activity
Service Implementation Activity
Associated Activities
Result Artifact

Fig. 2. Proposed Framework for Microservices Migrations

Phase 1: System Comprehension starts with a set of activities aiming to comprehend the existing system and assess alternatives as described by Wolfart et al. [16]. We depicted the common activities and involved personas. The activities in this phase are commonly performed as part of an architecture review using methods like ATAM, SAAM, or a more lightweight method, e.g., the one suggested by Auer et al. [2]. The resulting quality assessment links the decision for or against a migration to microservices to distinct scenarios and associated quality requirements which the architectural styles in question are favorable for.

Phase 2: Strategy Definition entails the two activities described in Sect. 1.1 to define the migration strategy. Depending on the system's technological state, organizational aspects or other boundary conditions, different strategies may be chosen. The selection of a suitable approach and technique for service identification depends on several factors like targeted quality attributes, the available input artifacts, automation potential and maturity of available tool support. To this end, academia offers a variety of approaches that partly offer freely available tools that can be leveraged by practitioners.

Phase 3: Architecture Definition starts with the identification of services and a preliminary definition of the target architecture. The incremental implementation of the identified services is preceded by a prioritization step. The framework puts a major focus on quality assurance aspects to ensure that initially defined measures are applied and satisfied. Hence, the implementation activities are accompanied by a scenario-based analysis and followed by a verification & validation step. Deviations from the targets will consequently lead to altering the defined target architecture or even considering an alternative service identification approach by stepping back into phase 2.

5 Current Status of Tool Support

To address RQ2, we are in the process to develop a web-based tool that guides architects through a migration scenario. In the comprehension phase, the tool collects system specifications and guides through an architecture assessment. Based on an extensible repository of approaches for service identification and architectural refactoring, the tool will further assist in finding the appropriate technique for a specific system. As such, the repository provides selected approaches proposed by academia. In the implementation phase, the guidance will be realized in form of suggesting patterns and best practices associated with the targeted quality aspects for the migration. Methodology and tool support are currently being refined and evaluated within longitudinal industry case studies. For the framework's structure and automation capabilities, we conducted interviews among 9 software professionals. We also see potential for hosting the developed tool publicly and expanding it by functionality to incorporate user feedback or a rating system. In that regard, the framework and tool support could facilitate knowledge transfer not just from academia to industry but also vice versa, thereby contributing to close the gap highlighted in Sect. 1.2.

References

1. Abdellatif, M., et al.: A taxonomy of service identification approaches for legacy software systems modernization. J. Syst. Softw. **173** (2021)
2. Auer, F., Lenarduzzi, V., Felderer, M., Taibi, D.: From monolithic systems to microservices: an assessment framework. Inf. Softw. Technol. **137** (2021)
3. Bajaj, D., Bharti, U., Goel, A., Gupta, S.C.: A prescriptive model for migration to microservices based on SDLC artifacts. J. Web Eng. (2021)
4. Bogner, J., Fritzsch, J., Wagner, S., Zimmermann, A.: Microservices in industry: insights into technologies, characteristics, and software quality. In: 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), pp. 187–195. IEEE (2019)
5. Bozan, K., Lyytinen, K., Rose, G.M.: How to transition incrementally to microservice architecture. Commun. ACM **64**(1), 79–85 (2021)
6. Cojocaru, M.D., Oprescu, A., Uta, A.: Attributes assessing the quality of microservices automatically decomposed from monolithic applications. In: 18th International Symposium on Parallel and Distributed Computing. ISPDC 2019, no. 1, pp. 84–93 (2019)
7. Conway, M.: Conway's law (2018). https://melconway.com/Home/Conways_Law.html. Accessed 11 July 2022
8. Di Francesco, P., Lago, P., Malavolta, I.: Migrating towards microservice architectures: an industrial survey. In: Proceedings - 2018 IEEE 15th International Conference on Software Architecture. ICSA 2018, pp. 29–38 (2018)
9. Fritzsch, J., Bogner, J., Wagner, S., Zimmermann, A.: Microservices migration in industry: intentions, strategies, and challenges. In: IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 481–490. IEEE (2019)
10. Fritzsch, J., Bogner, J., Zimmermann, A., Wagner, S.: From monolith to microservices: a classification of refactoring approaches. In: Bruel, J.-M., Mazzara, M., Meyer, B. (eds.) DEVOPS 2018. LNCS, vol. 11350, pp. 128–141. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-06019-0_10
11. Jamshidi, P., Pahl, C., Mendonca, N.C., Lewis, J., Tilkov, S.: Microservices: the journey so far and challenges ahead. IEEE Softw. **35**(3), 24–35 (2018)
12. Ponce, F., Márquez, G., Astudillo, H.: Migrating from monolithic architecture to microservices: a rapid review. In: Proceedings of 38th International Conference of the Chilean Computer Science Society (SCCC 2019), Chile (2019)
13. Schröer, C., Kruse, F., Marx Gómez, J.: A qualitative literature review on microservices identification approaches. In: Communications in Computer and Information Science, vol. 1310, pp. 151–168 (2020)
14. Taibi, D., Lenarduzzi, V., Pahl, C.: Processes, motivations, and issues for migrating to microservices architectures: an empirical investigation. IEEE Cloud Comput. **4**(5), 22–32 (2017)
15. Vale, G., Correia, F.F., Guerra, E.M., de Oliveira Rosa, T., Fritzsch, J., Bogner, J.: Designing microservice systems using patterns: an empirical study on quality trade-offs. In: 2022 IEEE 19th International Conference on Software Architecture (ICSA), pp. 69–79. IEEE (2022)
16. Wolfart, D., et al.: Modernizing legacy systems with microservices: a roadmap. In: Evaluation and Assessment in Software Engineering, pp. 149–159. ACM, New York, NY, USA (2021)