Towards a Process for Migrating Legacy Systems into Microservice Architectural Style

Daniele Wolfart[1], Ederson Schmeing[1], Gustavo C.L. Geraldino[1], Guilherme L.D. Villaca[1], Diogo do N. Paza[1], Diogo C.P. Domingos[1], Wesley K.G. Assunção[2,1], Ivonei F. da Silva[1], Victor F.A. Santander[1]

[1]PPGComp – Western Paraná State University (UNIOESTE). Cascavel, Brazil.

[2]COTSI – Federal University of Technology - Paraná (UTFPR). Toledo, Brazil.

{danielewolfart, edersonschmeing, gclgeraldino, guidvillaca, diogopaganinidomingos, diogopazacvel}@gmail.com, wesleyk@utfpr.edu.br, {ivonei.silva, Victor.Santander}@unioeste.br

Abstract. *Microservice architectural style is a paradigm to develop systems as a suite of small and autonomous services, communicating thought a lightweight protocol. Currently, one of the most common ways of adopting microservice architectures is by the modernization of legacy monolith systems. The migration of a legacy system into a microservice architecture is not a trivial task. In addition, there is a lack of recommendation or guidelines on how to perform such process. In view of this, this paper presents a preliminary process for conducting the migration of legacy systems into microservice architectures. This process was defined by analyzing and discussing pieces of work on the topic. As result, we propose a process composed of eight steps, grouped in four phases, which we describe together with their common input and output.*

1. Introduction

The majority of industrial systems are long-lived applications, i.e., legacy system, usually having a decayed and degraded monolithic architecture [Lewis et al. 2003]. In order to remain competitive, these monolithic legacy systems must be modernize. Nowadays we can observe a trend on migrating legacy systems into microservice architectures [Knoche and Hasselbring 2018]. The microservice architectural style is a paradigm where a system is a suite of small and autonomous services that work together [Newman 2015]. The benefits of adopting microservices are: reduced effort for maintenance and evolution, increased availability of services, ease of innovation, continuous delivery, ease of DevOps incorporation, and facilitated scalability [Taibi et al. 2017].

We can find in the literature mappings and reviews in the topic of migrating legacy system into microservices [Ponce et al. 2019, Di Francesco et al. 2019], classification of refactoring approaches for this migration [Fritzsch et al. 2018], industrial reports and surveys with practitioners [Carvalho et al. 2019b, Fritzsch et al. 2019, Di Francesco et al. 2018]. These papers describe, respectively, approaches to conduct the migration, refactoring approaches for deal with implementation artifacts, and experience reports on how migrations were applied in practice. Despite the grown of interest in migrating legacy systems into microservices, in the literature, to the best of our knowledge, there is no study that presents recommendations/guidelines on how to perform the entire migration process, covering activities of legacy comprehension, architecture definition, execution of the migration, and microservices monitoring.

The goal of this paper is to define a preliminary process for the migration of legacy systems into microservice architectures. For the definition of such process, we analyzed and extensively discussed a set of studies describing migration processes. Our study (presented in details in Section 3) was conducted in a subject of a master course with six students and one professor. The resulting process (described in Section 4) is composed of eight steps, grouped in four phases. The phases are: comprehension of the monolithic legacy system, definition of the new architecture, execution of the transformation, and monitoring after the migration. In addition, we also provide the observed input and output for each step. The contribution of preliminary process defined in our study (differences from related work are discussed in Section 2) is to serve as basis for those ones envisaging the conduction of a migration process.

2. Related work

In this section, we discuss pieces of work that are related to our study. In special the studies presented in [Mayer and Weinreich 2018], [Balalaie et al. 2018], [Chen et al. 2018], [Hassan et al. 2017], [Taibi et al. 2017], [Carvalho et al. 2019b], and [Ahmadvand and Ibrahim 2016].

Ahmadvand and Amjad Ibrahim present a methodology for microservice decomposition from system requirements. Security and scalability requirements are input for the decomposition. This methodology is base for architectural design decisions [Ahmadvand and Ibrahim 2016]. This approach is related to the steps one, two, and three of our proposed process. Balalaie *et al.* describe fifteen patterns to migrate a monolithic system into a set of microservices considering the variation of several factors such as requirements, current situation and skills of team members. The set of migration patterns focus on several steps in our proposed process, although the authors do not guarantee the completeness of patterns repository to realize the migration [Balalaie et al. 2018]. Chen *et al.* use the data flow diagram (DFD) to capture the business requirements of a monolithic system, then, an algorithm combines the same operations with the same type of output data into a decomposable data-flow, finally, a function is run to identify microservices candidates from the decomposable DFD [Chen et al. 2018]. This approach is related to the steps two and three of our proposed process. Hassan *et al.* describe an architecture-centric approach to model microservice granularity. They extend the *ambient* concept [Ali et al. 2010] by introducing *microservice ambient* that models microservices and uses *aspects* to support changes in granularity at runtime. As a result, this approach can provide architectural modeling to aid the candidate solution for granularity adaptation [Hassan et al. 2017]. This approach is related to the steps three and four of our process.

Taibi *et al.* after performing a survey with practitioners that migrate monolithic systems to microservices, describe a process framework adopted by the practitioners during the migration. This framework contains activities to migrate an existing monolithic system to a microservice from scratch. There is also activities to implement new features as microservices to replace gradually the existing system. This approach is related to the all steps of our proposed process, although the manuscript does not present details [Taibi et al. 2017]. Carvalho *et al.* also conducted a survey with practitioners and describe adopted criteria to extract microservices on monolithic systems. Requirements, modularity, cohesion, coupling, database scheme, visual models, and reuse were criteria more important mentioned by practitioners participants of the study [Carvalho et al. 2019b].

This approach is related to the step one and three of our proposed process. Mayer and Weinreich define an approach for continuously extracting the architecture of REST-based microservice software systems. The approach retrieves static and dynamic data from different involved and distributed microservices to overview the software architecture and supervise it over time [Mayer and Weinreich 2018]. This approach is related to the step eight of our proposed process.

Our study presented in this paper differs from these mentioned approaches by describing an entire process, capturing common steps, input and output. Therefore, the proposed process in this paper has the goal broader. Despite of existing secondary studies such as [Di Francesco et al. 2019, Fritzsch et al. 2018, Ponce et al. 2019, Francesco et al. 2017], which map primary studies describing characteristics of the migration, they do not describe a consolidated and comprehensive process. So, to the best of our knowledge, our study is the first one that presents recommendations and/or guidelines on how to perform the migration process.

3. Study Design

This section presents details of the methodology of our study.

3.1. Participants

The study was conducted by six master students and supervised by one professor, which is a researcher on the topic of this study. The students were enrolled in a course of Requirements Engineering from the Graduate Program X[1] at University Y[1]. The professor had experience on conducting research on migrating legacy systems to microservice architectures. As part of the course, the professor taught four classes, summing 3h30min, about the migration of legacy systems to microservices. These classes were designed to serve as a background for the students. In addition, the professor assisted the students and participated in all the activities of the study.

3.2. Primary Sources

The source of information to define the preliminary process were obtained from searches on Google Scholar[2] with different combinations of the keywords "Migration", "Microservices", "Legacy Systems", and "Monolith". The primary sources are: [Balalaie et al. 2018, Chen et al. 2018, Taibi et al. 2017, Ahmadvand and Ibrahim 2016, Mayer and Weinreich 2018, Hassan et al. 2017]. These papers were selected based on their content, which is strictly related to the goal of our work.

3.3. Data Extraction

Next we describe the steps we followed to collect relevant information from the papers and the discussions performed to define the preliminary process.

1. **Individual reading**: each paper was assigned to one student, which had one week to read the paper and answer the following questions: "*What are the driving forces to adopt a microservice architecture?*", "*What are the advantages and disadvantages of adopting microservices?*", and "*What does the paper describe about migrating a legacy system to a microservice architecture?*".

[1]Omitted due to double blind review.

[2]`https://scholar.google.com/` on November 12th, 2019.

2. **Individual presentation and group discussion**: the students individually presented their answers for the previous questions. During the presentation, all other students and the professor could ask questions, complement affirmations, or relate the information presented with information collected by other student. This activity took 3h30min.
3. **Papers filtering and pair reading**: after the individual presentations, we selected only three papers that were more related to the topic of migrating legacy systems to microservices. These papers were: [Balalaie et al. 2018, Chen et al. 2018, Taibi et al. 2017]. The professor defined pairs of students and assigned one of the selected papers to each pair. The pairs were asked to answer the following questions: "*Which are the steps to perform the migration of legacy systems to microservices?*" and "*What are the common input and output artifacts of each step?*". They had one week to read the papers and answer the questions.
4. **Pair presentation**: each pair of students presented the steps they found in the paper, as well as the input and output for each step. A open discussion was also allowed, similarly to Step 2. For the presentations and to support the discussion, we used a whiteboard, that was split in three parts, where each pair of students could draw the information they collected. Figure 1[3] presents the whiteboard after the presentation of the three pairs of students.
5. **Information merging**: the last step of our study was merging all the information presented by the students and discussed in group. For this, we relied on the drawing on the whiteboard (Figure 1). During the discussion for merging the initial processes found by each pairs, the students were asked to take notes. The resulting process after merging all information, is described in the next section.

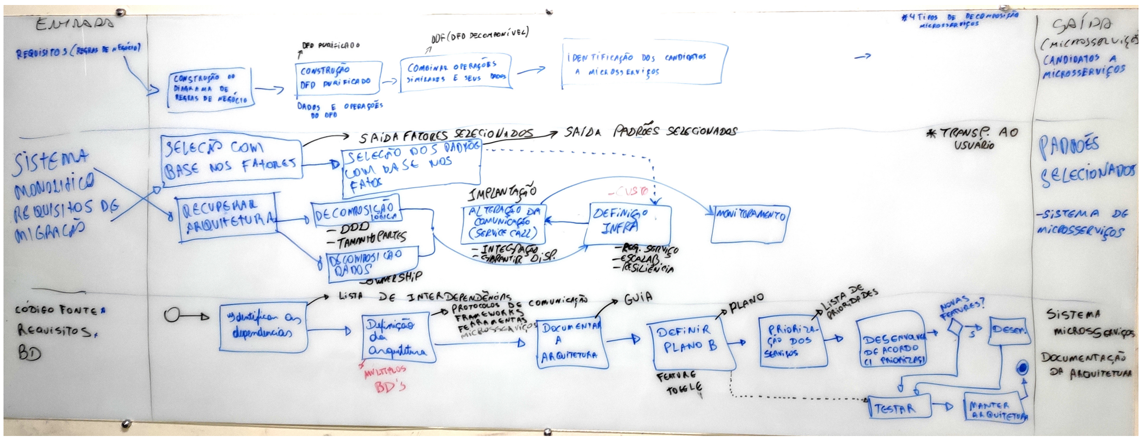

Figure 1. Whiteboard with the migration process identified the pairs of students.

4. Proposed Preliminary Process

After conducting the study presented in the previous section, we defined a preliminary process for migrating monolith legacy systems to microservice architectures. This preliminary process has four phases and a total of eight steps. Figure 2 presents the defined process, together with the input and output of each step. We stress here that this is a high-level process, and each step can be decomposed in lower level steps/tasks. In the next subsection we describe each step in details.

[3]The text in the whiteboard is in Portuguese, the language used in the course.

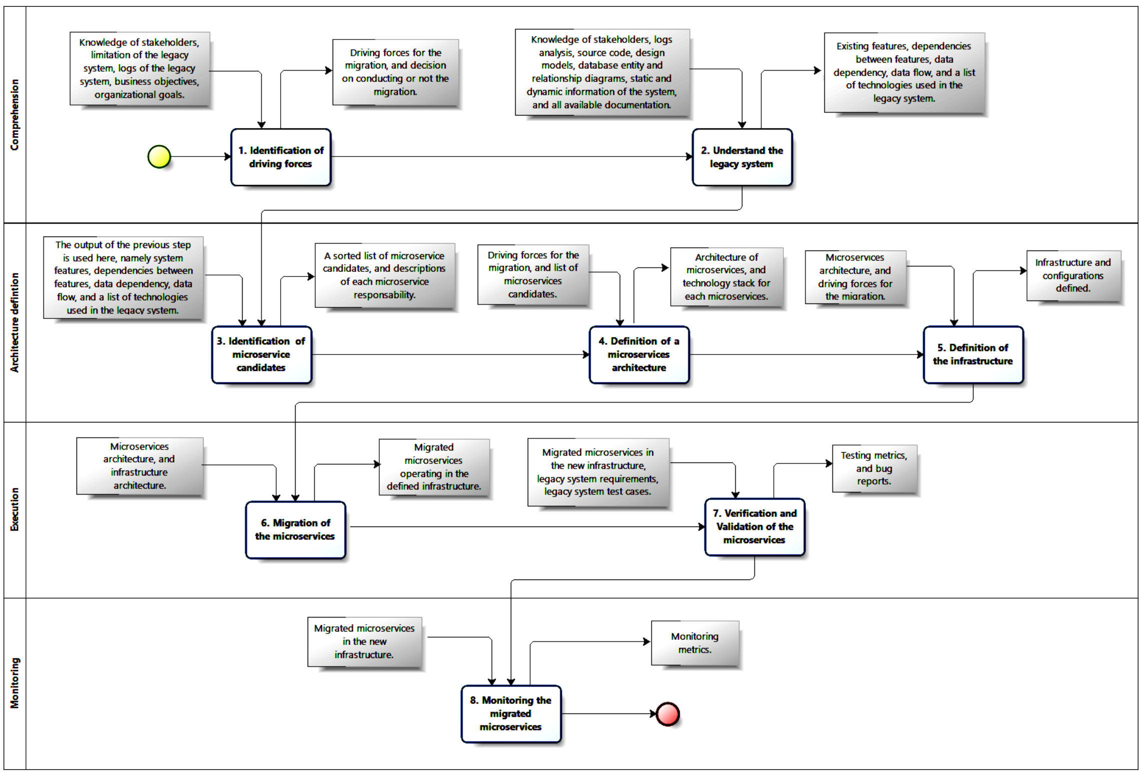
Comprehension
Knowledge of stakeholders, limitation of the legacy system, logs of the legacy system, business objectives, organizational goals.
Driving forces for the migration, and decision on conducting or not the migration.
Knowledge of stakeholders, logs analysis, source code, design models, database entity and relationship diagrams, static and dynamic information of the system, and all available documentation.
Existing features, dependencies between features, data dependency, data flow, and a list of technologies used in the legacy system.
1. Identification of driving forces
2. Understand the legacy system
Architecture defintion
The output of the previous step is used here, namely system features, dependencies between features, data dependency, data flow, and a list of technologies used in the legacy system.
A sorted list of microservice candidates, and descriptions of each microservice responsability.
Driving forces for the migration, and list of microservices candidates.
Architecture of microservices, and technology stack for each microservices.
Microservices architecture, and driving forces for the migration.
Infrastructure and configurations defined.
3. Identification of microservice candidates
4. Definition of a microservices architecture
5. Definition of the infrastructure
Execution
Microservices architecture, and infrastructure architecture.
Migrated microservices operating in the defined infrastructure.
Migrated microservices in the new infrastructure, legacy system requirements, legacy system test cases.
Testing metrics, and bug reports.
6. Migration of the microservices
7. Verification and Validation of the microservices
Monitoring
Migrated microservices in the new infrastructure.
Monitoring metrics.
8. Monitoring the migrated microservices

Figure 2. Process for migrating legacy systems to microservice architecture.

4.1. Phase 1: Comprehension the monolithic legacy system

This phase is responsible for providing a understanding of the legacy system and the needs on migrating it to a microservice architecture, i.e. the driving forces for the migration.

- **Step 1: Identification of driving forces**. The goal of this step is to identify the forces, requirements, needs, or objectives that motivate the migration. These forces can be related to technological, organizational, and/or business aspects. For example, in the primary sources we observe driving forces for the migration related to easier maintainability, allow scalability, easier management of development teams, moving towards adoption of DevOps [Taibi et al. 2017]. Other authors mention as motivation the opportunity to use of different technologies, reduced time-to-market, and better modularization [Balalaie et al. 2018].
  - *Input*: knowledge of stakeholders, limitation of the legacy system, logs of the legacy system, business objectives, organizational goals.
  - *Output*: driving forces for the migration, and decision on conducting or not the migration.
- **Step 2: Understand the legacy system.** This step is dedicated to an analysis of the actual legacy system. The goal is to understand the source code organization, existing features, dependencies among features, database structure, and all available documentation. During this step, the responsible for the migration also need to consider: (i) the business characteristics and goals such as keeping the company innovative and competitive; (ii) existing technologies related to programming languages, database management system, and software tools used; and (iii) organization aspects such as size of development teams, maturity and expertise of the

developers, development process adopted in the company, etc. The legacy system must be well-understood. In addition, the behavior of the systems and data flow within the features either [Chen et al. 2018].

  - *Input*: knowledge of stakeholders, logs analysis, source code, design models, database entity and relationship diagrams, static and dynamic information of the system, and all available documentation.
  - *Output*: existing features, dependencies between features, data dependency, data flow, and a list of technologies used in the legacy system.

4.2. Phase 2: Definition of the new architecture

The new architecture depends on what was chosen as migration goal. An incremental migration or big bang migration. In the former case, the architecture will be hybrid, having both the legacy system parts and the new microservices that will replace other of its parts. In the case of a big bang migration, an entire new microservice architecture should be defined. In addition to the architecture of the system under migration, the engineers need to define the infrastructure where the microservices will operate.

- **Step 3: Identification of microservice candidates**. This step has the goal of decomposing the legacy system in units that will be transformed in microservices. At first, the engineers should define the tasks each microservices will be responsible for, indirectly defining the size of the microservices in the new architecture [Hassan et al. 2017]. Based on the importance of defining the responsibility and size of a microservice, this step is a crucial for the migration process. The identification of microservice candidates can be performed by investigating the coupling and cohesion among parts of the legacy system, considering the migrating whole features to a microservice architecture, or focusing on migrate most reused parts of the legacy [Carvalho et al. 2019b]. A factor that must be taken into account during the identification of potential microservices is the overhead in communication that will be includes when decoupling to dependent task. In a monolithic system, all the communication among units are based on direct memory access. However, microservices use the network for the communication, having great impact in the time execution. For an incremental migration, Balalaie et al. recommend starting with a fewer number of microservices and incrementally perform the migration of new microservices, as the system grows and/or the development team acquire a better understanding of microservices characteristics [Balalaie et al. 2018].
  - *Input*: the output of the previous step is used here, namely system features, dependencies between features, data dependency, data flow, and a list of technologies used in the legacy system.
  - *Output*: a sorted list of microservice candidates, and descriptions of each microservice responsibility.
- **Step 4: Definition of a microservice architecture**. For example, here engineers have to describe the APIs for the microservices, define the communication protocol, and choose a strategy of services discovering. The architecture will be greatly affected by the decision of conducting an incremental or a big bang migration. In case of an incremental migration, the decisions on how integrate the microservices with legacy systems should be defined. For that, we observe the use of

Feature Toggles [Carvalho et al. 2019a], which ease to revert the use of the legacy system in case of any problem with the new microservices, allowing a better transition. Choosing a big bang migration, engineers have to figure out how to move all the legacy to microservices at once. A challenge with the big bang migration is to define a complete architecture beforehand. During the construction of this entire architecture some maintenance in the legacy during the migration should be updated in the new architecture, delaying the migration [Casey et al. 2017]. Another aspect that must be taken into account is the microservicification of the data. Commonly, legacy systems rely in one central database [Taibi et al. 2017]. But in the microservices context engineers can migrate the data of a microservice handle to a specific database managed only by this microservices and all access for such data is done by API requests. A central shared database also can be kept, however, this lead to data coupling among microservices, making complex activities of maintenance. In addition, it also affects the interdependence among different teams on charge of specific microservices.

  – *Input*: driving forces for the migration, and list of microservices candidates.
  – *Output*: architecture of microservices, and technology stack for each microservices.

- **Step 5: Definition of the infrastructure**. This step is devoted to define the environment where the microservices will operate. Here the engineers should decide about having a local hosts for the microservices or rent a cloud platforms such as Amazon Web Services[4], Microsoft Azure[5], or Google Cloud[6]. The infrastructure must provide resiliency and high availability in microservices, allowing management of scalability.
  – *Input*: microservice architecture, and driving forces for the migration.
  – *Output*: infrastructure and configurations defined.

4.3. Phase 3: Execution of the transformation

This phase is devoted to execute the migration considering all the planning of previous phases.

- **Step 6: Migration of the microservices**. Here the transformation of the implementation artifacts is done to move the parts of the legacy system to microservices. The infrastructure should be configured and be available for deploying the microservices. As we aforementioned, feature toggle[7] can be used as strategy to incrementally integrate the microservices with the legacy system, in a incremental migration.

  – *Input*: microservice architecture, and infrastructure architecture.
  – *Output*: migrated microservices operating in the defined infrastructure.

[4]`https://aws.amazon.com/`

[5]`https://azure.microsoft.com/`

[6]`https://cloud.google.com/`

[7]Feature Toggle is basically a flag variable used in a conditional statement with code blocks. Their goal is either enabling or disabling the feature code in those blocks [Rahman et al. 2016]

- **Step 7: Verification and Validation of the microservices**. After moving parts of the legacy systems to a microservice architecture, regression tests must be performed in order identify possible bugs. We highlight the importance of having proper test cases before starting the migration. Such tests are intend to identity bugs introduced during the migration and also confirm if the requirements of the migrated parts are adequate. Once interdependent microservices are migrated, integration tests are also required. For this step of verification and validation of the migration, the legacy system can be used as oracle for the testing activity.
  - *Input*: migrated microservices in the new infrastructure, legacy system requirements, legacy system test cases.
  - *Output*: testing metrics, and bug reports.

4.4. Phase 4: Monitoring after the migration

Monitoring is responsible to assess the health of the migrated microservices in the new infrastructure.

- **Step 8: Monitoring the migrated microservices**. The goal here is to keep control of the behavior of the migrated microservices in the new infrastructure. The monitoring should focus on availability of the services, bottlenecks, performance, use of infrastructure resources, and the like. Since a desired scalability is one of the main driving forces for migrated to a microservice architecture, here the engineers should analyze if this goal has been achieved. Loris Degioanni[8] describes five principles of monitoring microservices: (i) monitor containers and what runs inside them, (ii) use orchestration systems, (ii) prepare for elastic, multilocation services, (iv) monitor APIs, and (v) map monitoring to the organizational structure.
  - *Input*: migrated microservices in the new infrastructure.
  - *Output*: monitoring metrics.

5. Limitation of our study

Three limitations need be highlighted about our study, as follows:

- *Number of primary sources:* the process was based on some six primary studies. There are others that could be considered. In this case, we can only claim the process as a preliminary version based on relevant primary studies.
- *Process validation:* the proposed process was not evaluated in practice, then we cannot affirm that the process strictly adhere to practical needs and industrial scenarios. However, we alleviate this limitation by comparing the proposed process with information available in survey with practitioners, namely considering the studies [Taibi et al. 2017, Carvalho et al. 2019b, Carvalho et al. 2019a]. In addition, we believe the process was based on relevant literature for the field.
- *Expertise of the authors:* despite the professor that supervised this study being a researcher on the topic of this study, a threat to validity is the expertise of the students on migrating legacy systems to microservice architecture. To mitigate such threat, the professor provided a background training and assisted all steps of the study. In addition, the professor participated in all discussions regarding the primary sources.

[8]Available at `https://thenewstack.io/five-principles-monitoring-microservices/` accessed on March 11th, 2020.

6. Final Remarks

The microservice architectural style has emerged in industry[9] and only in the recent year has been focus of research in academia. Therefore, there is still scarce literature on the process of migrating monolith legacy systems to microservice architectures.

Most of existing studies on migrating legacy systems to microservices have focus specific scenarios or tasks of the migration process. Based on the lack of a broader view of the entire process, this paper describes a preliminary process composed of four phases that together have eight steps. As part of our preliminary process, we also described the input and output for each step. This proposal claims to be a starting point to consolidate a process of migrating monolithic systems into microservices.

Based on the limitations of our study, we plan future work based on mainly two directions. Firstly, we have been conducting an extend study including new primary sources to provide a more in-depth discussion of each step of the migration process. We intent to provide a process based on more primary studies to ratify or rectify the identified phases and steps of our preliminary process and increase new elements such as stakeholders, artifacts, guidelines and templates. Secondly, to move into an appropriate process that will be accepted and adopted by the practitioners, we intend to evaluate/validate the proposed process in real world case studies.

References

Ahmadvand, M. and Ibrahim, A. (2016). Requirements reconciliation for scalable and secure microservice (de)composition. In *IEEE 24th International Requirements Engineering Conference Workshops (REW)*, pages 68–73.

Ali, N., Ramos, I., and Solís, C. (2010). Ambient-prisma: Ambients in mobile aspect-oriented software architecture. *Journal of Systems and Software*, 83(6):937 – 958. Software Architecture and Mobility.

Balalaie, A., Heydarnoori, A., Jamshidi, P., Tamburri, D. A., and Lynn, T. (2018). Microservices migration patterns. *Software Practice and Experience*, 48(11):2019–2042.

Carvalho, L., Garcia, A., Assunção, W. K. G., Bonifácio, R., Tizzei, L. P., and Colanzi, T. E. (2019a). Extraction of configurable and reusable microservices from legacy systems: An exploratory study. In *23rd International Systems and Software Product Line Conference - Volume A*, pages 26–31, New York, NY, USA. ACM.

Carvalho, L., Garcia, A., Assunção, W. K. G., de Mello, R., and de Lima, M. J. (2019b). Analysis of the criteria adopted in industry to extract microservices. In *7th International Workshop on Conducting Empirical Studies in Industry and 6th International Workshop on Software Engineering Research and Industrial Practice*, pages 22–29. IEEE.

Casey, A., Beeler, R., Fry, C., Mauvais, J., Pernice, E., Shor, M., Spears, J., Speck, D., and West, S. (2017). Strategies for migrating to a new experiment setup tool at the national ignition facility. *JACoW Publishing*, pages 1–4.

Chen, R., Li, S., and Li, Z. (2018). From Monolith to Microservices: A Dataflow-Driven Approach. In *Asia-Pacific Software Engineering Conference (APSEC)*, pages 466–475.

[9]`https://martinfowler.com/articles/microservices.html`

Di Francesco, P., Lago, P., and Malavolta, I. (2018). Migrating towards microservice architectures: an industrial survey. In *IEEE international conference on software architecture (ICSA)*, pages 29–2909. IEEE.

Di Francesco, P., Lago, P., and Malavolta, I. (2019). Architecting with microservices: A systematic mapping study. *Journal of Systems and Software*, 150:77–97.

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

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

Fritzsch, J., Bogner, J., Zimmermann, A., and Wagner, S. (2018). From monolith to microservices: a classification of refactoring approaches. In *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*, pages 128–141. Springer.

Hassan, S., Ali, N., and Bahsoon, R. (2017). Microservice Ambients: An Architectural Meta-Modelling Approach for Microservice Granularity. *IEEE International Conference on Software Architecture (ICSA)*, pages 1–10.

Knoche, H. and Hasselbring, W. (2018). Using microservices for legacy software modernization. *IEEE Software*, 35(3):44–49.

Lewis, G., Plakosh, D., and Seacord, R. (2003). *Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*. Addison-Wesley Professional.

Mayer, B. and Weinreich, R. (2018). An approach to extract the architecture of microservice-based software systems. In *IEEE Symposium on Service-Oriented System Engineering (SOSE)*, pages 21–30.

Newman, S. (2015). *Building Microservices*. O'Reilly Media, 1st edition.

Ponce, F., Márquez, G., and Astudillo, H. (2019). Migrating from monolithic architecture to microservices: A rapid review. In *38th International Conference of the Chilean Computer Science Society (SCCC)*, pages 1–7. IEEE.

Rahman, M. T., Querel, L.-P., Rigby, P. C., and Adams, B. (2016). Feature toggles: Practitioner practices and a case study. In *13th International Conference on Mining Software Repositories (MSR)*, pages 201–211, New York, NY, USA. ACM.

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