2021 Second International Conference on Information Systems and Software Technologies (ICI2ST) | 978-1-6654-0411-2/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICI2ST51859.2021.00027

Monoliths to microservices - Migration Problems and Challenges: A SMS

Victor Velepucha
Facultad de Ingeniería de Sistemas
Escuela Politécnica Nacional
Quito, Ecuador
victor.velepucha@epn.edu.ec

Pamela Flores
Facultad de Ingeniería de Sistemas
Escuela Politécnica Nacional
Quito, Ecuador
pamela.flores@epn.edu.ec

***Abstract*—There are currently many applications that companies have accumulated over the years to leverage the operation of their businesses, however, over the years these applications have fallen into obsolescence. Most of these applications have been built following a monolith architectural style. With today's great technological advances and for companies to remain competitive, many of these applications need to be migrated and/or modernized to take advantage of current technologies. The process of migrating from a monolithic application to a microservice is a challenging process and there are few reported successful experiences. The objective of this paper is to contribute to the findings of problems and challenges reported in scientific literature, that companies have faced when migrating a monolithic application to a microservices architecture.**

***Index Terms*—microservices, monolith, migration process, microservices problems, microservices challenges**

I. Introduction

The traditional way of creating applications has been with monolithic architecture, that is, their deployment is as a single block. Some companies, as part of the evolution of their monolithic applications, are working on migrating them to an SOA architecture and others to a microservices architecture [1], [2]. With the birth of the microservice paradigm, DevOps practices and the use of Cloud Computing having also emerged, which have been adopted by companies such as Netflix, Amazon, and Uber, to create robust applications that contain microservices that can be developed, successfully deployed, tested, relying on the use of cloud platforms and containers [3], [4].

Currently there is a trend to modernize and migrate business applications towards a microservices architecture, and in this process problems have been encountered, challenges in which professionals from academia and industry have had to face [5], [6].

It is worth mentioning that a monolithic architecture has advantages and disadvantages just like an architecture in microservices, so before carrying out a migration, certain considerations must be made, such as analyzing the application to be migrated, to determine if a migration is worth doing.

The purpose of this paper is to conduct a literature review to identify the problems and challenges that arise during the migration process, and present this information in such a way that it serves as a point of reference for professionals who want to carry out this migration process.

This article is organized as follows: Section 2 presents general information about both architectures, Section 3 shows related concepts, Section 4 indicates the research method used, along with the research question and search strategy. Section 5 indicates how the search and data extraction were carried out. Section 6 discusses the findings. Finally, in Section 7 conclusions and future work are presented.

II. Background

A. Monolithic Architecture

An application with monolithic architecture is one that was built in such a way that its deployment is done in a single block. For example, an application built with an architecture of three logical layers, in the end, the deployment is done as a single block. A typical case of this architecture is a web application, which consists of the presentation, business and data access layer, whose objective is to separate and better organize the code, however, in execution, it acts as if it were a single block [7].

Monolithic architecture has been the traditional way of creating applications until today, however, this architecture has advantages and disadvantages. To name a few advantages, monolith architecture is easier to develop, easier to perform functional tests. Among the disadvantages; the complexity to incorporate changes, when a modification is made there is a high risk of introducing errors which can make the entire application unavailable, also slow delivery times, scaling is expensive, and there is complexity to adopt new technologies [7]–[12].

B. Microservices Architecture

An application that is built under microservice architecture follows the standard that each business functionality is trapped in a microservice, in such a way that there is only one responsibility for each microservice, and the recommendation is that each of these microservices consult your own database [13]. Microservice architecture is now adopted by many of the big companies like Amazon and Netflix [3], [4], it also has its advantages and disadvantages. Among the advantages, each microservice is independent, so if it occurs only this service is affected, ease of scaling, ease, and speed in development, given its small size, ease of using recent technology. However,

978-1-6654-0411-2/21/$31.00 ©2021 IEEE
DOI 10.1109/ICI2ST51859.2021.00027

this architecture also has its disadvantages, such as greater complexity as the number of microservices grows, more complexity in performing functional and technical tests, because a request can jump through several microservices, which can also lead to greater response times and higher latency on the network [7], [8], [12], [14], [15].

C. When to migrate a monolith to microservices?

There are recommendations that should be taken into account before carrying out the migration process. For example, if a monolithic application has little business functionality and is small, it is perhaps more convenient to keep it as is [16] and only perform an update to the most recent versions of the framework, version of the programming language, code optimizations. However, if an application is too large and complex to maintain, incorporating changes is too slow, the company needs the application to use recent technology [6], these are symptoms that the application must be modernized, and a viable alternative is to migrate to microservice architecture, despite having challenges to face, but at the same time many benefits.

Performing a migration process is a challenging process [6], since there is no single recipe to carry out this process, and several considerations must be made, such as deciding whether to perform a migration from scratch, or to perform an incremental migration in parts. The challenge in the migration process is to identify and isolate the business functionality and translate this functionality into a microservice. In Figure 1 shows how to carry out this process:

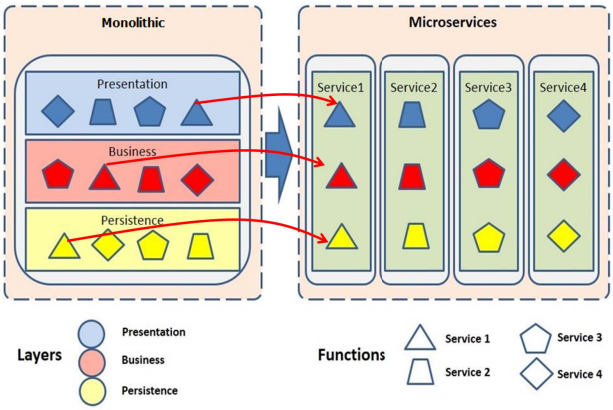

Fig. 1. Migration Process. (Source: Martin Fowler) [13]

III. Related concepts

When we talk about microservices, there are several related concepts, such as Cloud Computing, SOA, DevOps, Containers, Agile, which we present below.

- Service Oriented Architectures

Service Oriented Architecture (SOA) is a new style of information technology architecture that is based on the design of multiple services that can collaborate with each other to provide a final set of capabilities. Communication between the created services occurs through calls across the network instead of calls to internal methods of a process [17]. SOA emerged to facilitate communication between large monolithic applications. The design and implementation of an SOA architecture is governed by principles described in the SOA Manifesto, among which is promoting the reuse of software, facilitating the maintenance, or rewriting of software. SOA at its heart is a very sensitive idea, guaranteeing its discovery, being prepared to be used in compositions [18]. Instead, microservices are a modern interpretation of the SOA architecture, using simple communication protocols like HTTP with REST or lightweight messaging like RabbitMQ [19].

- Cloud Computing

Cloud computing is the delivery of computing services like servers, storage, databases and networking over the internet. With cloud computing we have faster innovation, flexibility in using resources and economies of scale, because you pay for only the services you use. Using cloud computing we have benefits like cost optimizations, efficient use of infrastructure, rapid enabling services when business demands [20]. Cloud-native architectures have emerged as a target platform for the deployment of microservice architectures [21]. Cloud services have some characteristics and considerations and all of them are explained in Table I:

Characteristic	Description
High availability	The ability to keep services running for long time periods with short downtime cycles.
Scalability	The ability to increase or decrease resources when business demands according to their current workload. You can add additional resources (scaling out) or additional capabilities (scaling up).
Elasticity	The ability to automatically or dynamically increase or decrease resources when business demands. Resources can be added or removed automatically; all of this can be done automatically.
Agility	The ability to react quickly. Resources can be allocated or deallocated in minutes and there is no manual intervention in provisioning or deprovisioning this service.
Fault tolerance	The ability to remain an up-and-running service even in the event of a component or service being no longer functional, this can be done with redundancy, then if one component fails the backup component takes its place.
Disaster recovery	The ability to recover all services from an event which has taken down all the cloud services. This can be done with geographic replication.
Global reach	The ability to reach audiences around the world twenty four hours, seven days a week, using the cloud infrastructure.
Customer latency capabilities	The ability to deploy some cloud services in specific datacenters around the world, solving some latency issues.
Predictive cost considerations	The ability to predict cost for using a particular cloud service and also perform analysis based on future business growth.
Security	The ability to provide better security than most organizations can achieve. Your data, applications, infrastructure are protected from potential threats.

TABLE I
CLOUD SERVICES CHARACTERISTICS.

It is possible to bring monolithic applications to computing using virtual machines, where the operating system runs the application and takes it to a virtual machine under IaaS mode (Infrastructure as a Service) [6]. This, however, involves costs that can be high, and the cloud provider may no longer support an outdated operating system. With the advances in the use of Cloud Computing, it is possible to isolate an application to work in containers, which is more economical than having a virtual machine.

- Containers

A container is an isolated user space environment where certain business functionality is executed. It is a way to modularize a system into parts that are much easier to manage and maintain, while also being resistant to failure [22]. Containers are a similar concept to virtualization, but a container is lighter; they are less demanding on resources and time, which is why they have been suggested as a solution for more interoperable application packages in the cloud [23]. There are benefits to the use of containers, such as; the security of having an application isolated in a container, hardware independence, use of management mechanisms and resource monitoring. Another benefit is the ability to develop, test and deploy applications on a large number of servers, as well as the ability to interconnect containers [23]. There are tools that have already been successfully tested for the use of containers, the most famous being Docker, which can package an application and its dependencies in a virtual container that can then be run on any Linux server. It also includes tools to create and manage clusters, manage persistent storage, build and share containers [22]. Among the benefits of Docker, is the flexibility and portability where the application can be executed, since the application placed in a container can be deployed on a physical server (On-Premise) or in Cloud Computing.

- Agile Software Development

It refers to software engineering methods for software development in an iterative and incremental way, where requirements evolve over time according to the need of the project. Thus, the work is carried out through the collaboration of self-organized and multidisciplinary teams, immersed in a shared process of short-term decision-making. The goal of each iteration is to deliver part of the required software functionality that works without errors, so that the customer can start using this functionality early [24], [25]. There are some methods and frameworks for agile software development, among the most prominent are; Scrum, Crystal Clear, extreme programming, adaptive software development (feature-driven development). Agile Software Development came about because organizations needed to release their software faster and more frequently. As they tried to overcome the strain this placed on their release management processes, they had to adopt patterns such as application release automation, leveraging DevOps practices.

- DevOps

DevOps is an acronym for development and operations, which refers to a software engineering practice that focuses on communication, collaboration, and integration between software developers (Dev) and systems professionals in information technologies (IT) in charge of the operation of the Software (Ops). The main characteristic is to reduce the time between the confirmation of a change in a system and the change that is placed in normal production, for which they rely on automation and monitoring tools, while ensuring high quality software delivery [26]. The relationship that exists between DevOps and microservices is that they are two practices whose purpose is based on offering maximum agility and operational efficiency to the organization, allowing optimization of developments, and leaving behind monolithic architecture. There are a few sets of DevOps tools that support the development and delivery process, such as Continuous Integration Tools (CI), Continuous Delivery (CD), Infrastructure as Code (IaaC).

IV. Research Method

To carry out this research were used the guidelines given by Barbara Kitchenham et al. [27] which are defined as: Research question, search for relevant information from literature, review of the most important articles found, classification of articles and extraction of relevant information.

A. Research question

For this study, the following research question was posed: RQ1: "What problems and challenges are there for the migration process of monolithic applications to microservices?"

The findings found are shown in the Table III, which presents the problems and challenges that developers face when carrying out this migration process.

B. Search strategy

As a strategy, an exploratory search is conducted, driven by the research question, based on the following search terms:

- **Search terms**
  Defined search criteria:
  - microservices migration
  - microservices problems
  - microservices challenges
  - monolithic microservices
  - monolithic modernization
- **Inclusion and exclusion criteria**
  As inclusion criteria:
  - Papers published from the year 2014 and beyond
  - Information related only with the Software Engineering area
  - Only journals, magazine, and conference papers

  As exclusion criteria:
  - Only papers written in English
  - Only journals, magazine, and conference papers
- **Literature resources**
  The scientific literature search was conducted with the most relevant databases such as: Scopus, IEEE Xplorer, ACM DL, Springer and Science Direct (SD).

Once the search strategy has been defined, the search strings are defined and refined, having the following search strings shown in Table II:

TABLE II
SEARCHES TERMS IN DATABASES.

Database	Search term
Scopus	(TITLE(microservices AND migration) OR TITLE(microservices AND problems) OR TITLE(microservices AND challenges) OR TITLE(monolithic AND microservices) OR TITLE(monolithic AND modernization))
IEEE	((("Document Title":microservices AND migration) OR "Document Title":microservices AND problems) OR "Document Title":microservices AND challenges) OR "Document Title":monolithic AND modernization)
ACM	[[Publication Title: microservices] AND [Publication Title: migration]] OR [[Publication Title: microservices] AND [Publication Title: problems]] OR [[Publication Title: microservices] AND [Publication Title: challenges]] OR [[Publication Title: monolithic] AND [Publication Title: microservices]] OR [[Publication Title: monolithic] AND [Publication Title: modernization]]
Springer	"microservices AND migration" OR "microservices AND problems" OR "microservices AND challenges" OR "monolithic AND microservices" OR "monolithic AND modernization"
SD	"microservices migration" OR "microservices problems" OR "microservices challenges" OR "monolithic microservices" OR "monolithic modernization"

V. EXECUTION PHASE

In the exploratory study carried out, related information is found in each of the different scientific databases, for which an information filter process is carried out in order to obtain the most relevant information. The process followed is shown in Figure 2:

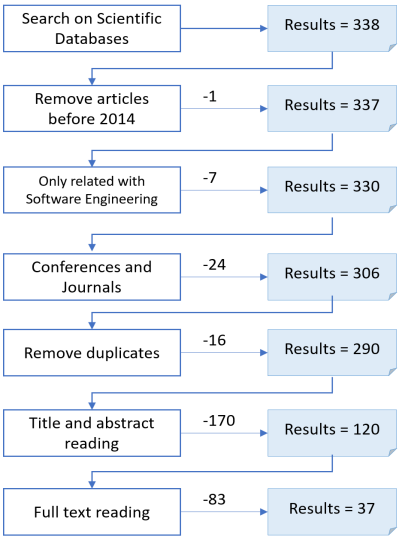

Fig. 2. Study Selection Process

A. Data extraction

As a result of this study, 37 final papers are obtained that show relevant information on problems and challenges to migrate applications with monolithic architectures to microservices. There are 23 conference papers (62 %) and 14 journal articles & magazines (38 %) as shown in Figure 3 and 4.

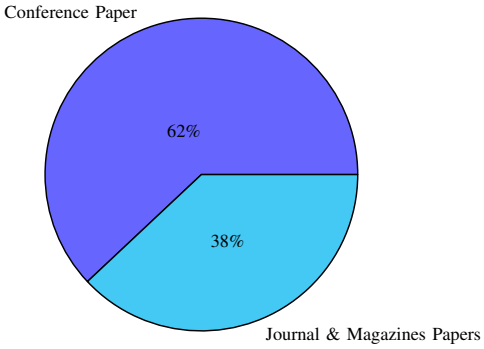

Fig. 3. % of conference papers and journals identified

From reading these papers, relevant information regarding problems and challenges can be found. Classification is carried out based on these main categories, and some of the articles can be repeated in more than one category:

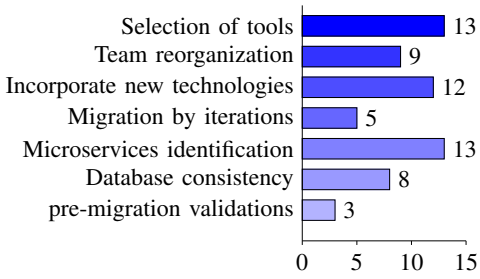

Fig. 4. Categories by problems and challenges

A classification by mixed, by group of problems, challenges and benefits is shown in Figure 5:

Once this classification has been carried out, a review of the problems and challenges is carried out and, if possible, contrasted with the information from other authors.

VI. DISCUSSION

From the relevant information found, below relevant information regarding the problems are presented by categories, challenges that are carried out during the process of migration from a monolith application to microservices. This same information is presented as a summary in the Table III.

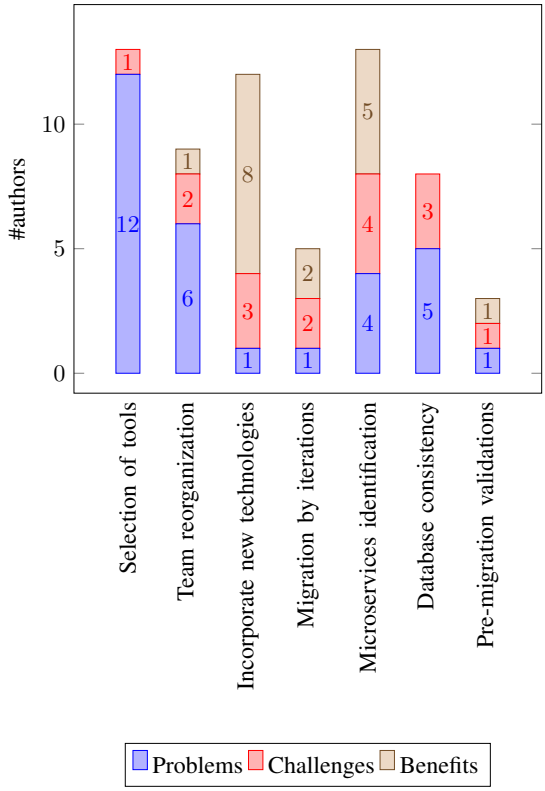

Fig. 5. Authors by categories and types

1) Use of support tools for the creation of microservices: There are tools on the market that only allow static analysis of an application code and generate a list of dependencies [2], [28], but this result is not enough to perform a correct decomposition of a monolith, that is why proposed framework proposals have emerged [29]–[31], models [32]–[34], methods [5], [35]–[37], that guide and support the migration of monolithic applications to microservices. For example, Cojocaru et al. [31] propose a framework for automatic validation of the decomposed microservices of a monolithic application, considering attributes such as granularity, coupling and cohesion between components, being an interesting proposal since it performs the validation of its framework with five applications.

2) Reorganization of work teams: One of the problems when migrating to microservices is the organizational change that an IT company faces. The organizational structure of most companies is hierarchical, and the responsibilities of the entire team are divided by technologies, where one team oversees a part, for example, one team is in charge of the front-end, another team of the database. When migrating to microservices, the challenge is to organize in a way that the different work teams are now responsible for services [8], [19], [29], [38]–[40]. Working under this new modality will take a learning curve [15], [41], however, the benefits will be seen in the medium and long term. By creating microservices that are loosely coupled, the responsible team can make better estimates of changes, they are adept at maintaining the services they are responsible for, and there is minimal impact on attrition as each team member is strong [5].

3) Incorporate the use of technologies and best practices: Part of the success in migrating a monolith to microservices is the selection and proper use of technologies and practices that have already been successfully tested in microservices. The recommendation is that the services use lightweight communication mechanisms, for example, presented as RestFul API [5]. Due to the highly distributed nature of microservices, as their number increases, they become more difficult to manage and monitor than monolith applications. That is why companies are challenged to incorporate the use of new technologies and practices to facilitate this task, such as the use of containers in Docker and administration of microservices with Kubernetes, DevOps tools [3], [6]. The challenge is not only in incorporating these technologies, but in training staff to make the correct use of them [5]. The great benefits of incorporating these technologies are in a better resilience, since if a microservice falls or is inhibited, another can be created immediately, scaling on demand and immediately, high numbers of compilations and versions per day through the use of Continuous Integration (CI) and Continuous Delivery (CD) [1], [3], [19], [38], [42].

4) Migrate to microservices in iterations: When you are in a process of migrating a monolith application in microservices, a strategy that can be very risky is rewriting all of the functionality in microservices in a single migration project [38], as this process can take a long time, in addition you run the risk of migrating functionality that is no longer used by the business, which is no longer necessary to migrate. The challenge is to properly plan and prioritize the business functionality to be migrated by holding meetings with expert business users [43]. The use of agile methodologies to perform iteration deliveries is a recommended practice to carry out this migration process [44]. The benefits of following this recommendation is that you would have the most used business functionality of the application early [15]. There is a technique to identify functional business flow following the Domain-Driven Design [38] which is one of the most recommended techniques.

5) Identify and properly design microservices: There are monolithic applications that were not created so they could be decomposed and migrated to a microservices architecture in the future. Therefore, the migration process can demand high costs and effort [45] and many times this process can be painful given the interdependencies between each of the components of the monolith [1], [5], so getting the right granularity for creating microservices becomes a challenging task. One of the strategies proposed by Zhang et al. [35] is to generate traceability logs in the monolith application, to be able to identify business functionality. Another challenge is to use the domain-driven design (DDD) pattern to be able to identify boundaries and create suitable candidate microservices that respond to a single business functionality, that have high

Problems/Issues	Challenges	Benefits/Solutions
Trying to perform the migration without using suitable tools.	Selecting a suitable tool, framework, method or model.	The migration process can be carried out in a standardized way and be a faster process.
When adopting microservices it is necessary to reorganize the work teams.	Change the hierarchical structure so that they now work based on microservices.	Best estimates to incorporate changes and speed to make changes.
Having a monolith makes it difficult to incorporate new technologies.	Relying on tools to automate most tasks, use cloud computing, containers, and staff training.	Speed in compilations, deployments, scalability, resilience.
Perform the migration to microservices complete and not in parts.	Prioritize business functionality to migrate, use agile methodologies.	Functionality that can be used early.
Problems to identify and design microservices.	Adapt the application to generate logs. Use design patterns such as DDD.	Reuse of microservices, speed in identifying where a failure occurs, ease of future migrations to new technologies.
Guarantee the consistency of the information when moving from a single database to multiple databases.	Lean on frameworks that facilitate and guarantee the consistency of the data.	By delegating responsibility for the consistency of information to frameworks or tools, programmers focus their efforts on developing the functionality required by the business.
Wanting to migrate all monolith applications to microservices.	Carrying out an adequate analysis of the application to be migrated and ensure good decision-making.	Optimize time, cost, resources.

TABLE III
COMPARATIVE OF PROBLEMS, CHALLENGES AND BENEFITS.

coupling and low cohesion. [5], [46], [47]. Another approach is to reverse engineer the application in such a way that the business functionality can be extracted, while trying to reduce the level of coupling in the microservices created [48]. For the design of microservices it is recommended that they be guided by the principles proposed by Fowler [13] and Newman [17]. Properly designing microservices brings the benefit that they can then continue to evolve towards new technologies, such as migrating to function-as-a-service (FaaS) [49]. Hassan et al. [50] propose a model called AMBIENT-PRISMA to identify functionalities of a monolith application to thus create each microservice, and in his research the challenge is to identify fine-grained business functionalities by boundaries and his contribution is to have a catalog of identified patterns. One of the benefits obtained by performing a correct modularization of microservices in reuse and ease of isolating specific points of failure, is independent development without dependencies between modules as if it occurs in monolithic applications [39]. Lotz et al. [39] identify as an opportunity the shorter time to market for creating microservices early, which allows a company to be more competitive. Taibi et al. [1] in their paper, highlights three important benefits of performing a migration; improvement in maintenance which leads to cost savings, ease of scalability and improvement in application performance.

6) Segmentation and consistency when dividing the database: Monolithic application generally leads to having a single database as a repository of information, and at the time of sending a request that involves several steps, these are executed by performing a commit or rollback, thus guaranteeing the consistency of the data. When migrating to microservices, the recommendation is to have one database per microservices [13]. Having a database per microservice, and having decentralized data, leads to the problem of guaranteeing the consistency of the data. To combat this there are proposals, such as the use of event-based messages, however, its implementation can be complex [5], [14], [40], [41], [51]. The challenge is to perform a correct division of the database, ensure effective sharing of the information, keep the information consistent and avoid performance problems. There are proposals in the literature, such as the proposals of Kholy et al. [30] who propose a framework to handle the transactionality between different databases. Relying on a framework prevents this programming logic from being carried out in the microservice. Fan et al. [36] propose a five-step method to improve data consistency in a microservices architecture, by means of a transaction coordinator, it ensures that each request is made correctly, and in case of an error, all of them are rolled-back transactions. A recent proposal [52] puts forward generating candidate microservices from the stored procedures of the database, which can be a way to follow, if the business rules are found in the stored procedures, which was common in applications from the 80's and 90's.

7) Pre-migration validations: Microservices is not always a Silver Bullet. As Brooks said [53], there is no technological development that alone promises an improvement in productivity, simplicity, reliability. Before performing a migration, you must validate how much functionality the monolith application has. If the application is small, with little business functionality and low complexity, the recommendation is to leave it as is, and only recompile the application towards the latest stable versions [16]. On the contrary, if the monolith application is large, and making changes is complex, it involves high maintenance costs, to incorporate new functionality, migrating to microservices is the recommended path [38]. Making the right decision allows you to optimize the use of resources, time, and cost.

A. Summary

The following Table III shows a summary of the problems and challenges encountered in the migration process between monolithic applications and microservices.

VII. CONCLUSION AND FUTURE WORK

In this research work, we discovered that each architecture, monolith and microservices has problems, challenges and

benefits, there is no single path or recipe towards migrating a monolithic application to microservices. Before making the decision to migrate an application with a monolithic architecture to a microservice architecture, it is recommended to review information found in the literature, to obtain experiences of how developers and researchers have carried out this process. In this research, models, frameworks, methods, tools used to carry out this process have been found, as well as providing relevant information about the benefits of each of the architectures. The research question has been answered: "What problems and challenges are there for the migration process of monolithic applications to microservices?", and our contribution is to provide a list of problems, challenges and benefits that arise when this process is carried out during migration. As future work, a detailed investigation will be carried out about the advantages and disadvantages of each of the architectures and case studies carried out of how to carry out the process of migration from a monolith application to a microservices architecture.

REFERENCES

[1] D. Taibi, V. Lenarduzzi, and C. Pahl, "Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation," *IEEE Cloud Computing*, vol. 4, no. 5, pp. 22–32, 2017.

[2] D. Taibi and K. Systä, "A decomposition and metric-based evaluation framework for microservices," *arXiv preprint arXiv:1908.08513*, 2019.

[3] D. Trihinas, A. Tryfonos, M. D. Dikaiakos, and G. Pallis, "Devops as a service: Pushing the boundaries of microservice adoption," *IEEE Internet Computing*, vol. 22, no. 3, pp. 65–71, 2018.

[4] M. Villamizar, O. Garcés, H. Castro, M. Verano, L. Salamanca, R. Casallas, and S. Gil, "Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud," in *2015 10th Computing Colombian Conference (10CCC)*. IEEE, 2015, pp. 583–590.

[5] H. Zhang, S. Li, Z. Jia, C. Zhong, and C. Zhang, "Microservice architecture in reality: An industrial inquiry," in *2019 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2019, pp. 51–60.

[6] J. Kazanavičius and D. Mažeika, "Migrating legacy software to microservices architecture," in *2019 Open Conference of Electrical, Electronic and Information Sciences (eStream)*. IEEE, 2019, pp. 1–5.

[7] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, "Microservices: yesterday, today, and tomorrow," in *Present and ulterior software engineering*. Springer, 2017, pp. 195–216.

[8] T. Prasandy, D. F. Murad, T. Darwis *et al.*, "Migrating application from monolith to microservices," in *2020 International Conference on Information Management and Technology (ICIMTech)*. IEEE, 2020, pp. 726–731.

[9] S. Newman, *Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith*. O'Reilly Media, 2019.

[10] J. Soldani, D. A. Tamburri, and W.-J. Van Den Heuvel, "The pains and gains of microservices: A systematic grey literature review," *Journal of Systems and Software*, vol. 146, pp. 215–232, 2018.

[11] D. Namiot and M. Sneps-Sneppe, "On micro-services architecture," *International Journal of Open Information Technologies*, vol. 2, no. 9, pp. 24–27, 2014.

[12] K. Gos and W. Zabierowski, "The comparison of microservice and monolithic architecture," in *2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)*. IEEE, 2020, pp. 150–153.

[13] M. Fowler and J. Lewis, "Microservices a definition of this new architectural term," *URL: http://martinfowler.com/articles/microservices.html*, p. 22, 2014.

[14] A. Furda, C. Fidge, O. Zimmermann, W. Kelly, and A. Barros, "Migrating enterprise legacy source code to microservices: on multitenancy, statefulness, and data consistency," *IEEE Software*, vol. 35, no. 3, pp. 63–72, 2017.

[15] G. Buchgeher, M. Winterer, R. Weinreich, J. Luger, R. Wingelhofer, and M. Aistleitner, "Microservices in a small development organization," in *European Conference on Software Architecture*. Springer, 2017, pp. 208–215.

[16] A. Singleton, "The economics of microservices," *IEEE Cloud Computing*, vol. 3, no. 5, pp. 16–20, 2016.

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

[18] Z. Xiao, I. Wijegunaratne, and X. Qiang, "Reflections on soa and microservices," in *2016 4th International Conference on Enterprise Systems (ES)*. IEEE, 2016, pp. 60–67.

[19] A. Bucchiarone, N. Dragoni, S. Dustdar, S. T. Larsen, and M. Mazzara, "From monolithic to microservices: An experience report from the banking domain," *Ieee Software*, vol. 35, no. 3, pp. 50–55, 2018.

[20] P. Mell, T. Grance *et al.*, "The nist definition of cloud computing," 2011.

[21] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices architecture enables devops: Migration to a cloud-native architecture," *Ieee Software*, vol. 33, no. 3, pp. 42–52, 2016.

[22] S. Grubor, *Deployment with Docker: Apply continuous integration models, deploy applications quicker, and scale at large by putting Docker to work*. Packt Publishing Ltd, 2017.

[23] C. Pahl, "Containerization and the paas cloud," *IEEE Cloud Computing*, vol. 2, no. 3, pp. 24–31, 2015.

[24] J. Shore *et al.*, *The Art of Agile Development: Pragmatic guide to agile software development*. " O'Reilly Media, Inc.", 2007.

[25] S. Nerur and V. Balijepally, "Theoretical reflections on agile development methodologies," *Communications of the ACM*, vol. 50, no. 3, pp. 79–83, 2007.

[26] L. Bass, I. Weber, and L. Zhu, *DevOps: A software architect's perspective*. Addison-Wesley Professional, 2015.

[27] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering–a systematic literature review," *Information and software technology*, vol. 51, no. 1, pp. 7–15, 2009.

[28] D. Taibi and K. Systä, "From monolithic systems to microservices: A decomposition framework based on process mining," in *8th International Conference on Cloud Computing and Services Science, CLOSER*, 2019.

[29] F. Auer, M. Felderer, and V. Lenarduzzi, "Towards defining a microservice migration framework," in *Proceedings of the 19th International Conference on Agile Software Development: Companion*, 2018, pp. 1–2.

[30] M. El Kholy and A. El Fatatry, "Framework for interaction between databases and microservice architecture," *IT Professional*, vol. 21, no. 5, pp. 57–63, 2019.

[31] M.-D. Cojocaru, A. Uta, and A.-M. Oprescu, "Microvalid: A validation framework for automatically decomposed microservices." in *CloudCom*, 2019, pp. 78–86.

[32] F. Rademacher, J. Sorgalla, and S. Sachweh, "Challenges of domain-driven microservice design: a model-driven perspective," *IEEE Software*, vol. 35, no. 3, pp. 36–43, 2018.

[33] Y. Yu, H. Silveira, and M. Sundaram, "A microservice based reference architecture model in the context of enterprise architecture," in *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*. IEEE, 2016, pp. 1856–1860.

[34] R. Petrasch, "Model-based engineering for microservice architectures using enterprise integration patterns for inter-service communication," in *2017 14th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. IEEE, 2017, pp. 1–4.

[35] Y. Zhang, B. Liu, L. Dai, K. Chen, and X. Cao, "Automated microservice identification in legacy systems with functional and non-functional metrics," in *2020 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2020, pp. 135–145.

[36] W. Fan, Z. Han, Y. Zhang, and R. Wang, "Method of maintaining data consistency in microservice architecture," in *2018 IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing,(HPSC) and IEEE International Conference on Intelligent Data and Security (IDS)*. IEEE Computer Society, 2018, pp. 47–50.

[37] A. Shimoda and T. Sunada, "Priority order determination method for extracting services stepwise from monolithic system," in *2018 7th International Congress on Advanced Applied Informatics (IIAI-AAI)*. IEEE, 2018, pp. 805–810.

[38] A. Carrasco, B. v. Bladel, and S. Demeyer, "Migrating towards microservices: migration and architecture smells," in *Proceedings of the 2nd International Workshop on Refactoring*, 2018, pp. 1–6.
[39] J. Lotz, A. Vogelsang, O. Benderius, and C. Berger, "Microservice architectures for advanced driver assistance systems: A case-study," in *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 2019, pp. 45–52.
[40] T. Salah, M. J. Zemerly, C. Y. Yeun, M. Al-Qutayri, and Y. Al-Hammadi, "The evolution of distributed systems towards microservices architecture," in *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, 2016, pp. 318–325.
[41] S. Baškarada, V. Nguyen, and A. Koronios, "Architecting microservices: practical opportunities and challenges," *Journal of Computer Information Systems*, pp. 1–9, 2018.
[42] R. Heinrich, A. van Hoorn, H. Knoche, F. Li, L. E. Lwakatare, C. Pahl, S. Schulte, and J. Wettinger, "Performance engineering for microservices: research challenges and directions," in *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion*, 2017, pp. 223–226.
[43] C. M. Aderaldo, N. C. Mendonça, C. Pahl, and P. Jamshidi, "Benchmark requirements for microservices architecture research," in *2017 IEEE/ACM 1st International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (ECASE)*. IEEE, 2017, pp. 8–13.
[44] T. Ueda, T. Nakaike, and M. Ohara, "Workload characterization for microservices," in *2016 IEEE international symposium on workload characterization (IISWC)*. IEEE, 2016, pp. 1–10.
[45] U. Zdun, E. Wittern, and P. Leitner, "Emerging trends, challenges, and experiences in devops and microservice apis," *IEEE Software*, vol. 37, no. 1, pp. 87–91, 2019.
[46] V. F. Pacheco, *Microservice Patterns and Best Practices: Explore patterns like CQRS and event sourcing to create scalable, maintainable, and testable microservices*. Packt Publishing Ltd, 2018.
[47] C.-Y. Fan and S.-P. Ma, "Migrating monolithic mobile application to microservice architecture: An experiment report," in *2017 IEEE International Conference on AI & Mobile Services (AIMS)*. IEEE, 2017, pp. 109–112.
[48] P. Di Francesco, P. Lago, and I. Malavolta, "Migrating towards microservice architectures: an industrial survey," in *2018 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2018, pp. 29–2909.
[49] E. Van Eyk, J. Grohmann, S. Eismann, A. Bauer, L. Versluis, L. Toader, N. Schmitt, N. Herbst, C. L. Abad, and A. Iosup, "The spec-rg reference architecture for faas: From microservices and containers to serverless platforms," *IEEE Internet Computing*, vol. 23, no. 6, pp. 7–18, 2019.
[50] S. Hassan, N. Ali, and R. Bahsoon, "Microservice ambients: An architectural meta-modelling approach for microservice granularity," in *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2017, pp. 1–10.
[51] R. M. Munaf, J. Ahmed, F. Khakwani, and T. Rana, "Microservices architecture: Challenges and proposed conceptual design," in *2019 International Conference on Communication Technologies (ComTech)*. IEEE, 2019, pp. 82–87.
[52] M. H. G. Barbosa and P. H. M. Maia, "Towards identifying microservice candidates from business rules implemented in stored procedures," in *2020 IEEE International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 2020, pp. 41–48.
[53] F. P. Brooks and N. S. Bullet, "Essence and accidents of software engineering," *IEEE Computer*, vol. 20, no. 4, 1997.