Analyzing Microservices and Monolithic Systems: Key Factors in Architecture, Development, and Operations

Juan Christian, Steven, Afdhal Kurniawan
Computer Science Department
School of Computer Science
Bina Nusantara University
Jakarta, Indonesia
{juan.christian, steven096, afdhal.kurniawan}@binus.ac.id

Maria Susan Anggreainy*
Computer Science Department
BINUS Graduate Program Doctor of Computer Science
Bina Nusantara University
Jakarta, Indonesia
maria.susan001@binus.ac.id

2023 6th International Conference of Computer and Informatics Engineering (IC2IE) | 979-8-3503-4516-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/IC2IE60547.2023.10331155

***Abstract*—Microservices and monolithic systems are two prevalent architectural approaches in software development. This study provides a complete review and analysis of the key components involved in design, development, and operation in software development. A systematic review of the literature was conducted according to the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) guidelines. This review examined various sources debating monolithic systems vs microservices, highlighting their benefits, drawbacks, and implementation in enterprises. The paper addresses important research questions, aiming to further analyze this architectural approach. Findings show that microservices offer benefits such as scalability, flexibility, and independent deployment, while monolithic systems provide simplicity and ease of development. However, challenges related to network communication, data consistency, and operational complexity were also found with microservices. This research focuses on discussing the trade-offs and factors to consider when deciding between monolithic systems and microservices, which provides in-depth information for practitioners in decision-making for software development. This research aims to help readers understand the effects of using monolithic or microservice-based systems in software development.**

Keywords—Microservices, Monolithic Systems, PRISMA, benefits, drawbacks, implementation

I. INTRODUCTION

In recent years, the approach to design a system has increased in variety. This could be seen from many kinds of companies and software engineers who apply various methods to system architecture. “Monolith” and “Microservice” are the two most prevalent high-level approaches to system architecture now [1].

Monolithic architecture is an approach where all application functionalities are kept inside a single codebase and all components are tangled together or closely coupled [2]. Smaller programs with constrained functionality may benefit from this approach, but when a program grows in complexity, it becomes more challenging to maintain, expand, and enhance. It is challenging to adapt and introduce new features since any change made to one part of the system can influence the entire system [1].

In contrast, microservice architecture as defined by Martin Fowler [3] is a modern approach in designing a software architecture where a single application is composed of many small and independent services that are loosely coupled and can be developed, deployed, and scaled independently. Independent services avoid SPOFs (Single Point of Failures), thus allowing other unaffected services to still work properly as intended [1]. However, these services offer a lot more complexity because of their independent nature, which needs a solid team to be able to successfully implement it.

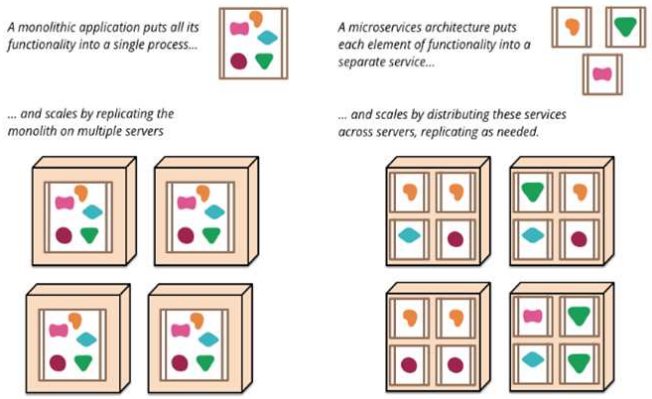

Fig. 1. Monoliths and microservices [3]

In order to give a thorough examination of microservices vs monolithic systems (can be seen in Fig. 1), this paper will focus on key aspects of architecture, development, and operations, in the intention of making it easy for practitioners to pick the best architecture for their system. For a complete examination, it is vital to compare between monolith and microservices in terms of its advantages, disadvantages, and key factors [4].

II. LITERATURE REVIEW

Microservices have been increasing in popularity in recent years as an alternative to traditional monolithic architecture. There are many big tech companies, such as Google and Amazon, use microservices for their applications. Microservice is an approach to software and systems architecture that builds on the well-established concept of modularization or collection of services but emphasizes technical boundaries [5], [6]. It introduces small services that are entirely deployable and scalable [7], which makes them easier to maintain, with less effort and more reliable with high availability [8].

There are many literatures works discussing microservices and comparing their performance to monolithic architecture. In research that was done by Al-Debagy and Martinek, the performance of microservices and monolithic application was compared. The results showed that the monolithic architecture performed better in terms of throughput in concurrency testing. However, there was no significant difference between the two architectures in load testing. In the study, the authors also assessed the performance of microservices applications utilizing various service discovery technologies. The findings revealed that applications employing Consul as the service discovery tool exhibited better throughput compared to other technologies. This highlights the importance of selecting an efficient service discovery solution, such as Consul, to optimize the performance of microservices applications [9]. There are many literatures works discussing microservices and comparing its performance to monolithic architecture. In research that was done by Al-Debagy and Martinek, the performance of microservices and a monolithic application was compared. The results showed that the monolithic architecture performed better in terms of throughput in concurrency testing. However, there was no significant difference between the two architectures in load testing. In their study, the authors also assessed the performance of microservices applications utilizing various service discovery technologies. The findings revealed that applications employing Consul as the service discovery tool exhibited better throughput compared to other technologies. This highlights the importance of selecting an efficient service discovery solution, such as Consul, to optimize the performance of microservices applications [10]. There is also a paper by J. A. Valdivia, X. Limón, K. Cortés-Verdín that contributes towards understanding of quality attributes and patterns in microservices by presenting a collection of microservice patterns and their association with quality attributes, The authors identified 21 microservice patterns and their benefits, trade-offs, and association with quality attributes [11].

Multiple approaches have been proposed for migrating from monolithic architecture to microservices architecture. One such approach, presented by Manabu Kamimura, Keisuke Yano, Tomomi Hatano, and Akihiko Matsuo, utilizes a software clustering algorithm called SArF to identify microservice candidates from the source code based on the relationships between "program groups" and "data" [12]. Another study by Francisco Ponce, Gastón Márquez, and Hernán Astudillo provides a comprehensive analysis of 20 migration techniques, highlighting the prevalence of design-based approaches and the predominance of object-oriented software and Java programming language in these proposals [13]. Additionally, Lorenzo De Lauretis proposes a strategy for automating the process of assigning business capabilities to microservices, although this research is still in the development phase [14]. These papers offer valuable insights into different strategies and techniques for successful migration to microservices architecture.

In their study, Miguel Brito, Jácome Cunha, and João Saraiva propose a novel approach for microservice identification from monolithic applications using topic modeling and clustering techniques. Their method leverages domain terms to identify services, resulting in a derived set of services based on the original software [15]. Another study presents an evidence-based decision support framework for organizations planning to migrate to microservices. The framework offers guidance on collecting and analyzing relevant characteristics and metrics to inform the redesign or rebuilding of monolithic systems. This framework proves valuable for companies seeking to migrate to microservices while mitigating the risk of overlooking critical information. It provides a comprehensive approach to ensure a successful and informed transition to microservices architecture [16]. Finally, a paper that focuses on decomposing monolithic software systems into microservices to be able to maintain them with less development effort. This paper can allow software developers to carefully use concrete guidelines in migrating their monolithic system into microservices [17].

But even though microservices architecture has been widely used, not all companies that use them have a deep understanding regarding the topic, which led some people to research it in [18], [19], [20]. A study conducted by Muhammad Waseem, Peng Liang, Mojtaba Shahin, Amleto Di Salle, and Gastón Márquez, aimed to deepen the understanding of microservices systems in the industry. The study suggests that more research is needed to face microservices design's complexity, security, monitoring and testing challenges through various number of solutions. The study also highlights the most used strategies, patterns, metrics, and practices for designing, monitoring, and testing microservices systems [21]. This is one of the reasons why this research is conducted.

The combination of DevOps and MSA can help organizations to achieve faster and more efficient software delivery by breaking down monolithic applications into smaller, independent services that can be developed, tested, and deployed independently. A study by Muhammad Waseem, Peng Liang, and Mojtaba Shahin identified several challenges and opportunities related to implementing DevOps in MSA, including the need for new tools and approaches to support the development and operation of microservices, the importance of communication and collaboration between development and operations teams, and the need for effective monitoring and management of microservices in production environments. Overall, the study suggests that the combination of DevOps and MSA has the potential to improve software delivery and quality in a wide range of application domains [22].

Microservices architecture has its own share of problems, such as its resource management, due to its larger configuration space compared to monolithic architecture. As such, a paper introduces PEMA (Practical Efficient Microservice Autoscaling), a lightweight microservices resource manager [23].

With everything that has been stated so far, it seems that microservices are a lot better than monolithic architecture, which is not true since some companies still use monolithic architecture. Some even moved from microservices to monolithic for several reasons, for example Istio. In the beginning, they thought that microservices would be the right approach from the very beginning, but in the process,

the benefits of using microservices didn't apply to their control plane, due to the components being installed and operated by a single team or individual. This, of course, led them to migrate their control plane from microservices to monolithic [24].

III. Research Questions

These are the questions in our research paper:

1. What are the key factors in the architecture, development, and operations of microservices and monolithic systems?
2. How do companies implement monolithic and microservices?
3. What are the advantages and disadvantages of both architectures?
4. What are the approaches to migrate from monolithic to microservices?

IV. Methodology

This study follows the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) guidelines (can be seen in Fig. 2). PRISMA provides a comprehensive and transparent approach to conduct systematic literature reviews, making sure that all the relevant studies are included. Before getting into PRISMA guidelines, inclusion and exclusion criteria of papers needed are important things to have (can be seen in Table I).

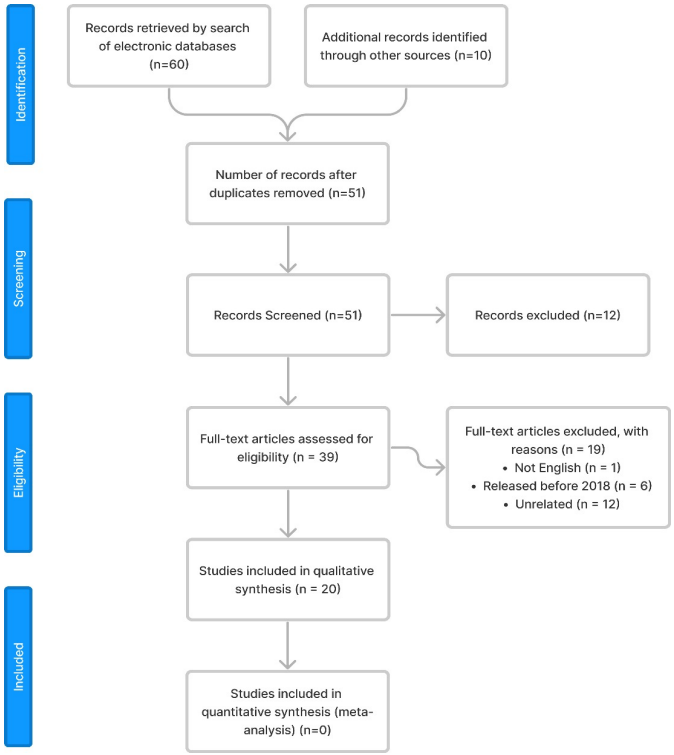

Fig. 2. PRISMA Flowchart

TABLE I. Inclusion and Exclusion Criteria

Inclusion Criteria	Exclusion Criteria
Papers from IEEE Xplore database, ACM Digital Library, Scopus, and Connected Papers	Papers not written in English
Papers released at a conference or in a journal	Papers released before 2018
Papers about key factors in architecture, development, and operations of microservices and monolithic systems	Papers unrelated to key factors in architecture, development, and operations of microservices and monolithic systems.

Search Strategy: To answer RQ.1 a search strategy was used to identify the eligibility and relevance of the studies addressing the key factors in architecture, development, and operations of microservices and monolithic systems. This includes checking database eligibility which we enquire papers from. Papers that we enquire about are from the selected electronic databases criteria. The search terms used included "microservices", "monolithic systems", "architecture", "development", and "operations".

To answer RQ.2 a similar search method was used to locate studies addressing the adoption of monolithic and microservices architectures in businesses. The electronic databases were searched using terms such as "implementation," "companies," "monolithic architecture," and "microservices architecture.". We also use multiple articles written by tech companies such as Netflix, Uber, and Grab.

The RQ.3 search approach attempted to find studies that discussed the benefits and drawbacks of both monolithic and microservices systems. The electronic databases were combed via utilizing terms like "advantages," "disadvantages," "monolithic architecture," and "microservices architecture."

Study Selection: After searching papers for microservices and monolithic system's key factors in architecture, development, and operations, then papers will be selected based on the criteria that have been made before. The inclusion criteria of this paper consist of papers from electronic databases, is a journal or released in a conference, and addresses key factors in architecture, development, and operations of microservices and monolithic systems. Papers that are not written in English, were released before 2018, and unrelated to research questions are excluded.

Data Extraction and Analysis: Identifying and obtaining relevant papers based on the study questions was part of the data extraction and analysis procedure. These studies' characteristics then were extracted: author(s), publication year, and research technique. Key findings and methodological details were also collected. The retrieved data were classified and categorized into themes or subjects related to the study questions throughout the data analysis phase.

V. Result and Discussion

RQ1: *What are the key factors in the architecture, development, and operations of microservices and monolithic systems?*

The literature review revealed several key factors in the architecture, development, and operations of microservices and monolithic systems. Firstly, microservices architecture emphasizes technical boundaries and introduces small,

deployable, and scalable services that are easier to maintain and offer high availability. On the other hand, monolithic architecture has been found to perform better in terms of throughput in concurrency testing. Additionally, studies have shown that vertical scaling is more cost-effective than horizontal scaling in the Azure cloud for monolithic systems.

Furthermore, the review identified 21 microservice patterns, 3 different types of microservice architectures that are Microservice Architecture (MSA), Service Oriented Architecture (SOA), and Cloud Architecture (CA). We also identified their association with quality attributes, providing insights into the relationship between architecture and system quality (can be seen in Table II).

TABLE II. MICROSERVICES PATTERNS

Pattern name	Architecture
API gateway	MSA
Asynchronous completion token	SOA
Asynchronous query	SOA
Auth-service	MSA
Backend for frontend	MSA
Change code dependency to service call/adapter microservice	MSA
Circuit breaker	MSA
Competing consumers	CA
Container	MSA
Correlation ID	MSA
Deploy cluster and orchestrate containers	MSA
Edge server	MSA
Event notification	SOA
External load balancer	MSA
Externalized configuration	CA
Gatekeeper	CA
Internal load balancer	MSA
Key-Value store	MSA
Load balancer/load-balancing	MSA and CA
Local database proxy	CA
Local sharding-based router	CA

RQ2: *How do companies implement monolithic and microservices?*

Most of the papers that were found discussed more about microservices than monolithic. Out of many studies that have been included in this paper, only one talks about a company's monolithic implementation, and that is Istio.

Istio is a system that provides some way to manage cloud-native applications that are deployed in Kubernetes. Istio is split into a data plane and a control plane. The data plane consists of envoy proxies that mediate communications between application containers. After the data plane comes the control plane which manages and configures envoy proxies inside the data plane to monitor and route application traffic based on the user-provided traffic control guidelines [24]. This architecture can be seen in Fig. 3.

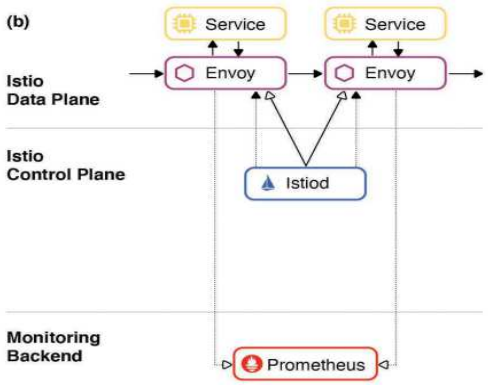

Fig. 3. Istio's monolithic control plane [24]

Istio's control plane, Istiod, is made up of the unification of its services. These are pilot, citadel, galley, injector, and mixer. This unification gives some benefits to Istio, such as simplified installation, easier configuration, and easier debugging since on most installations, it is operated by a single team or individual.

Other than Istio, there are some well-known companies that also use monolithic architecture, such as Stack Overflow and Amazon through its Prime Video [25]. Stack Overflow uses the components on their application efficiently, which helps in balancing server loads and lowers its latency [26]. Amazon Prime Video uses monolithic to avoid unnecessary expenditure [27]. Amazon, Netflix, and Spotify are setting a very fast trend in the microservices industry [28]. It is common for industries to implement microservices that save time and resources such as DevOps, Docker and auto-scaling [29].

In recent years, microservices have gained considerable attention and have been widely discussed. Many companies employ various combinations of APIs, web hosting services, and coding languages to construct their applications' microservices architecture. According to interviews conducted by Justus Bogner, Jonas Fritzsch, Stefan Wagner, and Alfred Zimmermann, it was discovered that all 14 interviewees, representing different companies utilizing service-based systems, rely on RESTful HTTP for their software communication and use Java for the language. Then, 11 out of 14 interviewees used Docker containers for service deployment.

Interviews conducted by Qilin Xiang, Xin Peng, Chuan He, Hanzhang Wang, Tao Xie, Dewei Liu, Gang Zhang, and Yuanfang Cai also revealed the same results regarding the software communications of the interviewee used. All of them rely on HTTP/REST, some combined with other communication styles such as RPC and Messaging [20] (can be seen in Table III).

TABLE III. MICROSERVICES COMMUNICATION USED [20]

Modes	#
HTTP/REST, Messaging	18
HTTP/REST	15
HTTP/REST, RPC, Messaging	12
RPC	4
HTTP/REST, RPC	2

RQ3: *What are the advantages and disadvantages of both architectures?*

The literature review highlighted various advantages and disadvantages of both microservices and monolithic designs. Microservices provide benefits such as simplified maintenance, scalability, and high availability. Microservices enable modular development and deployment by breaking down applications into smaller, independent services, allowing teams to work on multiple services at the same time [7], [8]. This modularity also makes maintenance and upgrades easier because changes to one service do not affect the entire system. Microservices can scale horizontally, allowing companies to accommodate rising traffic and demand by deploying more instances of individual services.

However, microservices bring additional complexity in managing distributed systems, as well as security problems and operational overhead in managing various services [23]. Monolithic architectures, on the other hand, have their own set of advantages and disadvantages. Monolithic systems frequently outperform in terms of throughput, particularly in concurrent testing settings [9]. Communication between modules is often faster than inter-service communication with microservices because all components execute in a single process. Because the entire system is packaged and distributed as a single unit, monolithic architectures are also relatively easier to create and deploy.

On the other side, monolithic systems lack the modularity and flexibility that microservices offer. A monolithic system frequently necessitates rebuilding and redeploying the entire application to accommodate changes or upgrades to one module. As a result, there may be longer development cycles and slower times to market for new features. Monolithic systems' individual parts might interconnect tightly over time, making it challenging to isolate and test them. Additionally, the inability to scale some modules independently may restrict the amount of independent scaling possible, leading to inefficient use of resources [23]. A summarized overview of the advantages and disadvantages is provided in the following Table IV.

TABLE IV. ADVANTAGES AND DISADVANTAGES OF MICROSERVICES AND MONOLITHIC ARCHITECTURE

Architecture	Advantages	Disadvantages
Microservices	Easier maintenance	Increased complexity
	Scalability	Security challenges
	High availability	Complex monitoring and testing requirements
		Operational overhead in managing multiple services
Monolithic	Better throughput	Lack of modularity and flexibility
	performance	
	Simpler development and deployment process	Longer development cycles and slower time-to-market
	Cost-effective vertical scaling	Tightly coupled components in the system
		Inability to scale specific modules independently

RQ4: *What are the approaches to migrate from monolithic to microservices?*

Many approaches are used to migrate from a monolithic to a microservice, such as Model-Driven, Static Analysis, Dynamic Analysis [13]. These approaches are used to identify which application could be migrated from monolithic to microservices, thus named microservices candidates. Model-Driven approach utilizes design elements, such as business capabilities. Static Analysis approach utilizes source code. Dynamic Analysis approach utilizes system functionalities during runtime.

Miguel Brito, Jácome Cunha, and João Saraiva present a technique in their paper that utilizes Static Analysis as a foundation and incorporates topic models [15]. The approach involves extracting information from the source code of the monolithic application, which is lexical and structural. This extracted data is then applied to a topic modeling technique, which assigns topics to different parts of the application. By combining the topic distribution with the structural information, the researchers employed a clustering algorithm to identify potential microservice candidates within the application. In summary, their methodology involves extracting and analyzing code information, applying topic modeling, and using clustering algorithms to pinpoint suitable microservice components within the monolithic system.

Besides these approaches, there is one more approach that is proposed by Lorenzo De Lauretis and that is an approach that is based on business functionality concepts [14]. As for the research, it consists of five stages, which can be seen in Fig. 4.

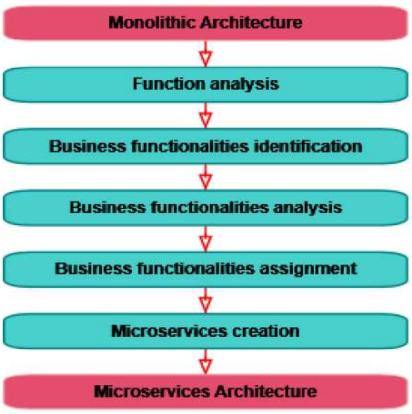

Fig. 4. Monolithic to microservices migration strategy based around business functionality concepts [14]

VI. Conclusion

Lastly, the literature study offers light on the thorough examination of microservices and monolithic systems, analyzing major architectural, development, and operational variables. It demonstrates that, while microservices provide benefits such as faster maintenance, scalability, and high availability, they also add difficulties to the management of distributed systems. Monolithic design, on the other hand, simplifies development and gives superior throughput performance, but it loses modularity and flexibility. Migrating from a monolithic system to microservices necessitates considerable preparation and thought. These insights can help firms make educated decisions about their architectural options and future migration tactics.

While previous research gives useful insights, more studies are required to address the complexity of microservices design, security concerns, monitoring and testing problems, and the effective implementation of DevOps in microservices architectures. Continued research in these areas will contribute to a better understanding and practice of microservices.

References

[1] Freddy Tapia, Miguel Angel Mora, et al. From Monolithic Systems to Microservices: A Comparative Study of Performance. Appl. Sci. 2020, 10(17), 5797; https://doi.org/10.3390/app10175797.

[2] G. Blinowski, A. Ojdowska and A. Przybyłek, "Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation," in IEEE Access, vol. 10, pp. 20357-20374, 2022, doi: 10.1109/ACCESS.2022.3152803.

[3] Martin Fowler. APIs should not be copyrightable. 16 December 2014.

[4] K. Gos and W. Zabierowski, "The Comparison of Microservice and Monolithic Architecture," 2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH), Lviv, Ukraine, 2020, pp. 150-153, doi: 10.1109/MEMSTECH49584.2020.9109514.

[5] P. Jamshidi, C. Pahl, N. C. Mendonça, J. Lewis and S. Tilkov, "Microservices: The Journey So Far and Challenges Ahead," in IEEE Software, vol. 35, no. 3, pp. 24-35, May/June 2018, doi: 10.1109/MS.2018.2141039.

[6] A. Selmadji, A. -D. Seriai, H. L. Bouziane, R. Oumarou Mahamane, P. Zaragoza and C. Dony, "From Monolithic Architecture Style to Microservice one Based on a Semi-Automatic Approach," 2020 IEEE International Conference on Software Architecture (ICSA), Salvador, Brazil, 2020, pp. 157-168, doi: 10.1109/ICSA47634.2020.00023.

[7] S. Eski and F. Buzluca, "An automatic extraction approach: transition to microservices architecture from monolithic application," in *Proceedings of the 19th International Conference on Agile Software Development: Companion*, 2018.

[8] O. Al-Debagy and P. Martinek, "A comparative review of microservices and monolithic architectures," May 2019.

[9] G. Blinowski, A. Ojdowska and A. Przybylek, "Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation," *IEEE Access,* vol. 10, p. 20357–20374, 2022.

[10] J. A. Valdivia, X. Limon and K. Cortes-Verdin, "Quality attributes in patterns related to microservice architecture: a Systematic Literature Review," in *2019 7th International Conference in Software Engineering Research and Innovation (CONISOFT)*, Mexico, 2019.

[11] J. A. Valdivia, X. Limon and K. Cortes-Verdin, "Quality attributes in patterns related to microservice architecture: a Systematic Literature Review," in *2019 7th International Conference in Software Engineering Research and Innovation (CONISOFT)*, Mexico, 2019.

[12] M. Kamimura, K. Yano, T. Hatano and A. Matsuo, "Extracting candidates of microservices from monolithic application code," in *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*, Nara, 2018.

[13] F. Ponce, G. Marquez and H. Astudillo, "Migrating from monolithic architecture to microservices: A Rapid Review," in *2019 38th International Conference of the Chilean Computer Science Society (SCCC)*, Concepcion, 2019.

[14] L. De Lauretis, "From Monolithic Architecture to Microservices Architecture," 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Berlin, Germany, 2019, pp. 93-96, doi: 10.1109/ISSREW.2019.00050.

[15] M. Brito, J. Cunha and J. Saraiva, "Identification of microservices from monolithic applications through topic modelling," in *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, New York, NY, USA, 2021.

[16] F. Auer, V. Lenarduzzi, M. Felderer and D. Taibi, "From monolithic systems to Microservices: An assessment framework," *Inf. Softw. Technol.,* vol. 137, p. 106600, September 2021.

[17] D. Kuryazov, D. Jabborov and B. Khujamuratov, "Towards Decomposing Monolithic Applications into Microservices," 2020 IEEE 14th International Conference on Application of Information and Communication Technologies (AICT), Tashkent, Uzbekistan, 2020, pp. 1-4, doi: 10.1109/AICT50176.2020.9368571.

[18] J. Bogner, J. Fritzsch, S. Wagner and A. Zimmermann, "Microservices in Industry: Insights into Technologies, Characteristics, and Software Quality," 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), Hamburg, Germany, 2019, pp. 187-195, doi: 10.1109/ICSA-C.2019.00041.

[19] X. Zhou, S. Li, L. Cao, H. Zhang, Z. Jia, C. Zhong, Z. Shan and M. A. Babar, "Revisiting the practices and pains of microservice architecture in reality: An industrial inquiry," *J. Syst. Softw.,* vol. 195, p. 111521, January 2023.

[20] Q. Xiang, X. Peng, C. He, H. Wang, T. Xie, D. Liu, G. Zhang and Y. Cai, "No free lunch: Microservice practices reconsidered in industry," June 2021.

[21] M. Waseem, P. Liang, M. Shahin, A. Di Salle and G. Márquez, "Design, monitoring, and testing of microservices systems: The practitioners' perspective," August 2021.

[22] M. Waseem, P. Liang and M. Shahin, "A Systematic Mapping Study on Microservices Architecture in DevOps," August 2020.

[23] M. R. Hossen, M. A. Islam and K. Ahmed, "Practical Efficient Microservice Autoscaling with QoS Assurance," in *Proceedings of the 31st International Symposium on High-Performance Parallel and Distributed Computing*, New York, NY, USA, 2022.

[24] N. C. Mendonça, C. Box, C. Manolache and L. Ryan, "The Monolith Strikes Back: Why Istio Migrated From Microservices to a Monolithic Architecture," in IEEE Software, vol. 38, no. 5, pp. 17-22, Sept.-Oct. 2021, doi: 10.1109/MS.2021.3080335.

[25] Christian de Looper. Everything you need to know about Prime Video. December 27, 2022.

[26] Navjot Bansal.Case Study: How Stackoverflow's monolith beats microservice performance. 18 April 2023.

[27] Michael Solati.Amazon Ditches Microservices for Monolith: Decoding Prime Video's Architectural Shift.May 18, 2023.

[28] Vincent Bushong., Amr S. Abdelfattah, et.al."On Microservice Analysis and Architecture Evolution : A Systematic Mapping Study." Appl. Sci. 2021, 11, 7856. https://doi.org/10.3390/app11177856.