International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 /doi:10.14445/22315381/IJETT-V70I3P225

Volume 70 Issue 3, 222-233, March 2022

Review Article

Taxonomical Classification and Systematic Review on Microservices

Sidath Weerasinghe[1], Indika Perera[2]

[1]*Research Scholar, Department of Computer Science & Engineering, University of Moratuwa, Sri Lanka.*
[2]*Professor, Department of Computer Science & Engineering, University of Moratuwa, Sri Lanka.*

[1]weerasingheldsb.20@uom.lk, [2]indika@cse.mrt.ac.lk

Abstract *- The software industry widely used monolithic system architecture in the past to build enterprise-grade software. Such software is deployed on the self-managed on-premises servers. Monolithic architecture systems introduced many difficulties when transitioning to cloud platforms and new technologies due to scalability, flexibility, performance issues, and lower business value. As a result, people are bound to consider the new software paradigm with the separation of concern concept. Microservice architecture was introduced to the world as an emerging software architecture style for overcoming monolithic architectural limitations. This paper illustrates the taxonomical classification of microservice architecture and a systematic review of the current state of the microservice architecture by comparing it to the past and future using the PRISMA model. Conference papers and journal papers the base on the defined keywords from well-known research publishers. The results showcase that most researchers and enterprise-grade companies use microservice architecture to develop cloud-native applications. On the contrary, they are struggling with certain performance issues in the overall application. The acquired results can facilitate the researchers and architects in the software engineering domain who aspire to be concerned with new technology trends about service-oriented architecture and cloud-native development.*

Keywords - *Microservices, Systematic review, PRISMA, Cloud computing, Architecture.*

I. INTRODUCTION

Monolithic software architecture is a traditional way of building enterprise-grade software. In this architecture, all the modules are encapsulated into one single package and deployed in one single server. Those applications are named self-contained software. All the modules that reside in the monolithic software are tightly coupled and could have thousands of different services in a single executable file. For example, can consider the web-based application consisting of the significant three ties: user interface, business logic, and the data access layer. From the standpoint of monolithic architecture, all the ties are written in a single code base and deployed in one package. Most of the architects focus on the concept of granularity in software architecture. Monolithic software architecture solely allows the coarse granularity level since monolithic architecture has significant subcomponents and single objects hold a lot of data in the entire system. The object-Oriented programming (OOP) concept is commonly used to develop software. Many programming languages and languages versions are released to the development community with the use of the OOP concept. With technological development, people are prone to embrace various other concepts for software architecture. As a result, 'separation of concerns' was introduced to the computer science field. As a consequence, people transited to Service Oriented Architecture (SOA). This architecture has become quite popular among software developers who develop software in distributed environments [1]. In the SOA architecture, services are the stand-alone components deployed on the distributed environment and they perform the service orchestration over the network. Service-Oriented Computing (SOC) is a concept that applies services as the underlying elements for creating applications [2]. Software engineers use the SOA principle and the SOC concept to develop the application rather than going for monolithic architecture.

Amidst the popularity of cloud computing, people are keener towards a developed application that possesses the capability of using cloud services and deploying on the cloud environment. That kind of application is called a cloud-native application. New technology trends are also introduced to the world for application development. Software architecture is influenced by cloud-native technologies as well [3]. As a result, microservice architecture is introduced to the world to overcome the issues of existing software architectures. In the microservice architecture, all the services are deployed in the distributed environment and necessary services are called for to satisfy the business requirements. Such services are minute software components and intend to logically perform a specific task. A lot of microservice frameworks such as Spring Boot, Vert.x, Go Fast HTTP, etc. are introduced to facilitate the microservice development. Hence, developers can gain more

advantages in using those software architecture frameworks because they enable additional features in the form of simple libraries such as security, service discovery, distributed tracing, etc.

The objective here is to identify the taxonomical classification and systematic comparison of the microservice-based research studies and their real-world usage. Hence, considered some selected papers from well-known scientific publishers to conduct this research. Clarified and compared the microservice architecture's different qualities and how they evolved from the earlier stage to the current state and then used current highly practiced research methodologies, techniques, research approaches, and methods to conduct the research mapping study on the microservice architecture. This research study reveals that microservice architecture has addressed the issues which persist in the monolith software architecture. Most of the microservice's quality attributes bring more value to the developers. But, certain areas like inter-service communications in the microservice architecture need to be further improved.

The outcomes of this research study benefit the researchers in the subject areas of software architecture and cloud computing. People who work in the capacity of software designers will assist this study in choosing the correct technology, tools, and methods for their projects. Through this study, persons who are looking for converting a monolithic application to the microservice architecture can gain ample ideas to make decisions.

II. BACKGROUND

Several review studies related to the microservice architecture were conducted in the past few years to leverage the trending software concepts. Most of the researchers conducted studies related to the microservice transition and contemplated the problems associated with it [4]. The optimal way of transformation in the microservice is to define the service on the monolithic systems as fine-grained services. Such services should be in the form of atomic services; which means that one service is responsible for performing only a specific task. A term called ‘microservitization’ is introduced to illustrate the system transition to the microservices. The main challenge in transforming applications to microservice architecture is to define the services as independent modules. Another systematic literature review was conducted by Atilim University with relation to microservices [5]. The study focused on three main areas such as types of research conducted related to the microservices, motivation behind microservice architecture, and the emerging trends in microservice architecture. They have classified microservice-based research into several classes such as validation research, evaluation research, solutions, philosophy, and experience. According to their research, most of the people performed the solutions research related to the microservice system development and its usage [5]. A relatively few studies are conducted on the philosophical area of the microservices as it is already defined by Lewis and Fowler [6]. Most of the research focuses on the microservices design and the functionalities. Recent microservice studies show that most of the microservice architectural concerns are overcome using continuous deployments and cloud containerization technologies [7].

A systematic grey literature review was conducted by the group of research related to the gap of the microservice architecture and also to the advantages of the microservice architecture [8]. According to them, the main pain area of the microservice design stage is the security policies definitions to the resource levels of the services. Handling the distributed storage and application testing on the distributed environment is mentioned as pains on the microservice architecture in the application development stage. In the operational time, the main pain point is the huge network consumption because of the inter-service communication in a microservice architecture. When developing the overall solution using the microservice architecture, all teams need to conclude what sort of communication mechanism should be used for the interservice communication.

Nowadays, most architects use cloud-native software architecture for the development of applications. According to the researchers, the best cloud-native software architecture is microservice architecture [9]. They comprise several non-functional requirements for the cloud-native applications such as elasticity, scalability, automated deployment, and vendor lock-in avoidance [10]. Docker the Rocket containerization concepts together with the Kubernetes. Docker Swarm and Mesos automated container management help microservice architecture to bring those non-functional requirements to the deployments.

A German university conducted research related to the features of the microservice architecture [11]. According to their research, security, performance resilience, reliability, latency, and fault tolerance are the most important features of the microservice architecture. Most people use the microservice architecture to get proper scalability, extensibility, and agility. With these intentions, engineers focus on the security of the microservices because it is deployed in the distributed environment. When services are deployed in remote locations, developers need to scrutinize the entire application performance and the response time.
Most of the researchers conduct the review using standards research approaches such as the quantitative research approach and the qualitative research approach. In the qualitative research approach, they collect feedback from the engineers who are involved in the microservice architectures via interviews, open-ended questionnaires, and surveys. As qualitative research approaches researchers build the prototypes of the microservices and collect the actual data for their research analysis.

In summary, identified that most of the industries use the microservices architecture for their upcoming developments and some of them are faced with difficulties in the microservice architecture. The key motivation of this study is to find out the encouragement of converting the old system to the microservice architecture and fill the gap of the microservice architecture by the current state of practice using the systematic review of the microservices.

II. RESEARCH METHOD

This systematic review methodology is conducted using the guidelines provided by the well know PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analysis) model. The primary aim of the systematic literature review is to identify the presented results related to the specific domain and then evaluate those results. Finally, illustrate the identified results in a meaningful manner.

A. Research Question

This systematic review aims to consolidate research findings that are conducted by several research institutes and technology companies related to the microservice architecture by answering the following research questions.

- What are the main motivations to convert the monolithic application to microservice architecture?
- What are the technologies & architectural patterns used in microservice architecture with technological advancement?
- What are the pains of people in the software development life cycle when using the microservice architecture?

The aim is to address the above-mentioned research questions by studying the past research activities, and in the meantime provide taxonomical classification regarding the microservice architecture. People who are still struggling with the monolithic systems can get a better advantage from this research to convert their systems into microservices architecture using improved technologies and the architectural patterns in their software development life cycle**.**

a) What are the main motivations to convert the monolithic application to microservice architecture?

In this scenario, intend to elaborate on the main motivations behind converting monolithic systems into a microservice architecture. The existing issues people face in the monolithic, and how they mitigate those problems via the microservice architecture will be further discussed in this section.

b) What are the technologies & architectural patterns used in microservice architecture with technological advancement?

There are several architectural patterns for microservice architecture. Here, aim to bring several architectural patterns for problem-solving in the microservice architecture, as well as discuss the new technological tools and frameworks that are used for microservice development. According to the application domain, developers need to change the tech stack to cater to the business requirements.

c) What are the pains of people in the software development life cycle when using the microservice architecture?

Noticed that some of the research studies elaborate on the problems associated with the microservice architecture in the software development life cycle. The aim is to identify those problems and provide possible solutions to mitigate those types of pains.

B. Data Source

IEEE Xplore [12], ACM [13], Citeseerx [14] and ScienceDirect [15] digital libraries were used for this research study. The reason behind choosing these digital libraries is due to high-quality research and easy accessibility. Simultaneously, the research databases support the advanced searching facility to search for any metadata of the publication. Out of those databases, chose the published research papers and journal papers for this study.

C. Search Strategy

Identification of the search term is a vital factor in systematic reviews and it is mandatory to use proper search terms in the relevant field to identify the research works.

Primary studies were conducted using high-impact digital libraries, and well-reputed conference proceedings were used to provide answers to the research questions. Search is bounded to the online library search engines which are available for university subscriptions. Time limits weren't applied for the search. Further to that, noticed that microservice research started in 2000 decade.

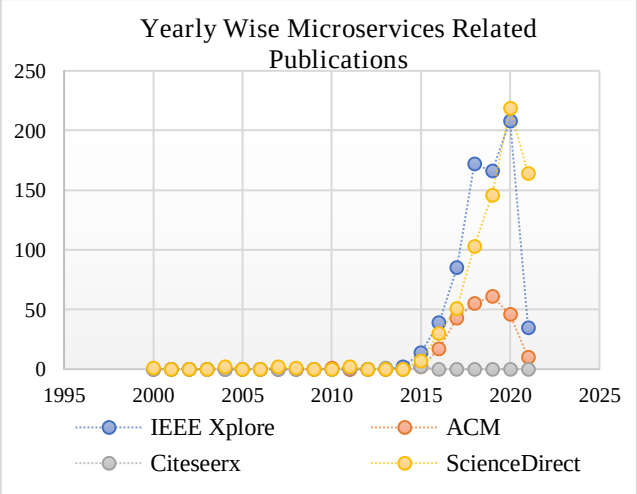

Fig. 1 Yearly wise publication count

Below combinations of strings were used for the search:

General Search string – ("Microservices" OR "Microservice" OR "micro-service") AND ("monolithic application to microservice architecture") OR ("microservice framework" OR "microservice development tools") OR ("microservice architectural patterns") OR ("Challenges of microservice")

According to the above graph 1, after 2016 microservice became a popular area for research studies. Last year research published a vast number of publications related to microservices.

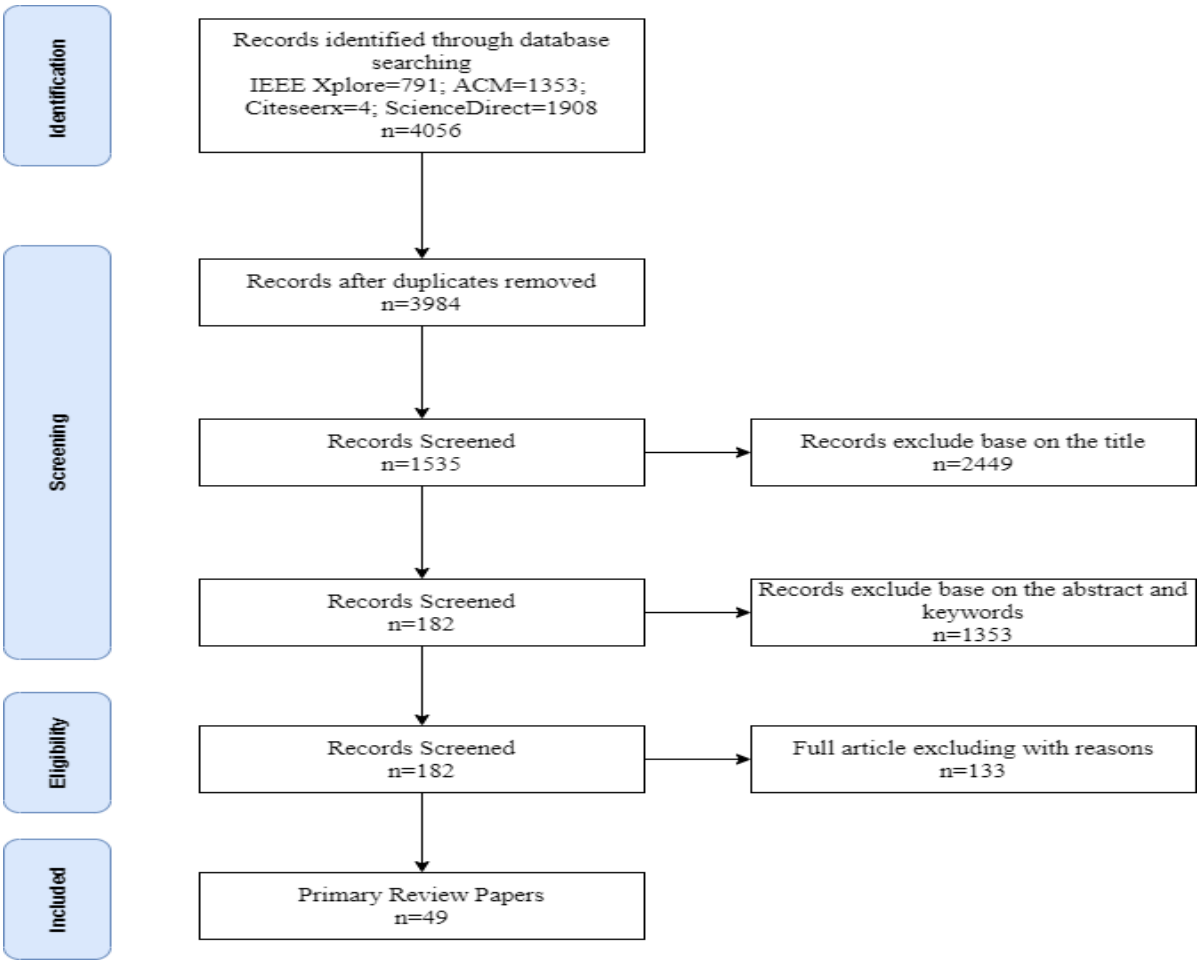

Fig. 2 PRISMA model

Once the papers related to the microservices were sorted out, once again categorized the papers according to each research, question-wise using the paper's keyword, title, and abstract. conducted each research question studies using the filtered output.

D. Study Selection Process

With adherence to the PRISMA guidelines, [16] selection process is carried out for the systematic review which is shown in the above figure 2. Standard tools like Microsoft excel and the Zotero tool are used to analyze and store the papers. In each phase conducted the paper selection and the filtering without being biased and made confidence in the study process.

E. Inclusion, Exclusion Criteria

Inclusion criteria are included from the data sources for this research study. Exclusion criteria that are used for the research studies are not relevant to the systematic review. These criteria are applicable for all data sources that have been used for the study.

- Inclusion criteria 1 – Research studies that are relevant to the search string from the data sources
- Inclusion criteria 2 – Research conference papers and Journal papers are considered
- Exclusion criteria 1 – Paper languages other than English

- Exclusion criteria 2 – Papers are not relevant to the research questions
- Exclusion criteria 3 – Research studies that are unclear to the domain

III. RESULTS AND DISCUSSION

A. What are the Main Motivations to Convert the Monolithic Application to Microservice Architecture?

Publication	Reference
Journal	[17], [18], [19]
Conference	[20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31]

In the 1990 decade, most software development companies developed enterprise-grade software using the monolith software architecture. Back in that period, client requirements are very unique and bound to a specific scope. Those requirements are not changing over time and possess a clear understanding of the requirements. When moving to the 2000 decade, it is depicted that the world is subject to technological advancement. Day by day new technology is introduced while the existing technology has improved. With such improvements in technology, clients' requirements have turned out to be more complex and advanced. To cater to those requirements people should need to move out from the traditional software architectures [32]. It can categorize people's intentions to choose the architecture as product, cost, and process.

When considering the product, most of the engineers focus on the product scalability over the cloud, product maintainability, performance, and as well as product security [33]. Most people are moving towards digital services, and service consumers tend to use digital services as well. Hence, application owners need to scale their applications. However, with the monolithic architecture, it is very costly, and they can't scale the required service only. Therefore, with the monolithic architecture, service providers can't compete with the news industry. Due to this fact, people are motivated to move their architecture to microservice-based architecture. Day to day requirements tends to change with the digitization of the services and there's an increased need to adhere to those changes as well as to introduce new features. In the monolith architecture, all the components are tightly coupled, and the change of one component may affect the entire application. The maintainability of the code is challenging in monolithic architecture due to the tightly coupled modules. In a microservice architecture, all the services work independently and are deployed separately. Hence, adding new features and changing the code is very easy and testability is very high [34]. Certain industries need a higher level of security while some of them need an acceptable level of security. The monolithic architecture contains some security modules, and most of the systems are having only one checking point. On the contrary, code-level security is also facing difficulties in changing some dependencies in the monolithic architecture. But in the microservice architecture, developers can implement several checkpoints to validate security. Software performance can be measured using several aspects such as response time, throughput, and software capacity. Most of the real-time systems look for less response time. On the other hand, they struggle with the microservice architecture as microservice architecture is deployed in the distributed environment, and service to service communication will add some latency to the response time [35].

But when we consider the throughput, microservice architecture provides massive throughput with architectural behavior. Single microservice can handle an extensive workload because of its independent behavior. Hence, microservice is better capacity-wise in comparison to the monolithic system. For such reasons, a lot of people move their systems towards a microservice architecture. Many companies initiate the process of frequent software releases. In the monolithic architecture, all the modules are tightly coupled to each other. Hence, adding new features or patching existing logic takes a lot of engineering effort. When converting that effort to the cost, it is considerably larger in comparison to project cost. In the actual business context, software companies need to provide solutions within the shortest time frame and the lowest cost to win over the business. That is very challenging in monolithic based systems. But with the microservice architecture, people can easily patch and adapt to the new requirements within short cycles with the leverage of Continuous Integration (CI) and Continuous Deployment (CD) pipelines [36]. With the microservice architecture, software companies can run an agile-based software development process that can cater to the frequent requirement changes.

B. What are the Architectural Patterns used in Microservice Architecture with Technological Advancement?

Publication	References
Journal	[37], [38], [39]
Conference	[40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58]

Microservice architecture is one of the emerging architectures in the software industry. With the motivation towards microservice architecture, software companies convert their monolithic-based software and service-oriented architecture software to microservice-based architecture. Several frameworks have been developed for microservice development in various programming languages. Those frameworks expose the interfaces to use and write the business logic to the users. Underline framework implementation helps establish several features to microservice such as security features, resiliency,

asynchronous programming, cloud enablement, etc. By choosing the framework, developers need to speculate several factors like framework maturity, in-build features, development support, performance, and software license. In consideration to all these factors, choose the most used and popular microservice development framework to perform the systematic reviews such as Java base Spring Boot framework [59], Go-based Go Micro framework [60], Node.js based Molecular framework [61], and Java bases, Vert. x framework [62].

Table 1. Microservice framework comparison

	Spring Boot	Go Micro	Moleculer	Vert.X
Founded Date	01.10. 2002	06.12. 2017	17.02 2017	10.10. 2011
Contributors	831	135	1721	212
Github Star	56800	16600	4600	12300
Releases	204	92	99	123
Available Documentation (1-5 Rating)	5	3	4	2
Stack Overflow Tags	25	2	2	10
Dependency Management	Maven, Gradle, Ant	Go Build	npm	Maven, Gradle, Kotlin

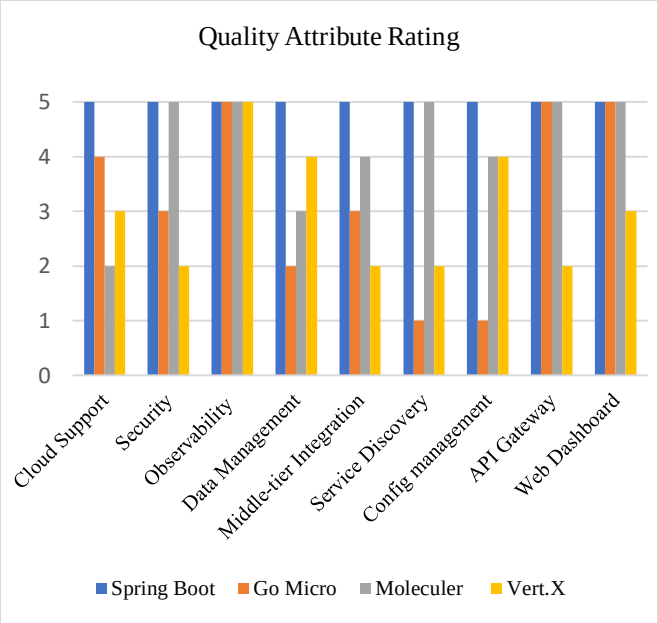

Fig. 3 Quality attributes ranking in microservice

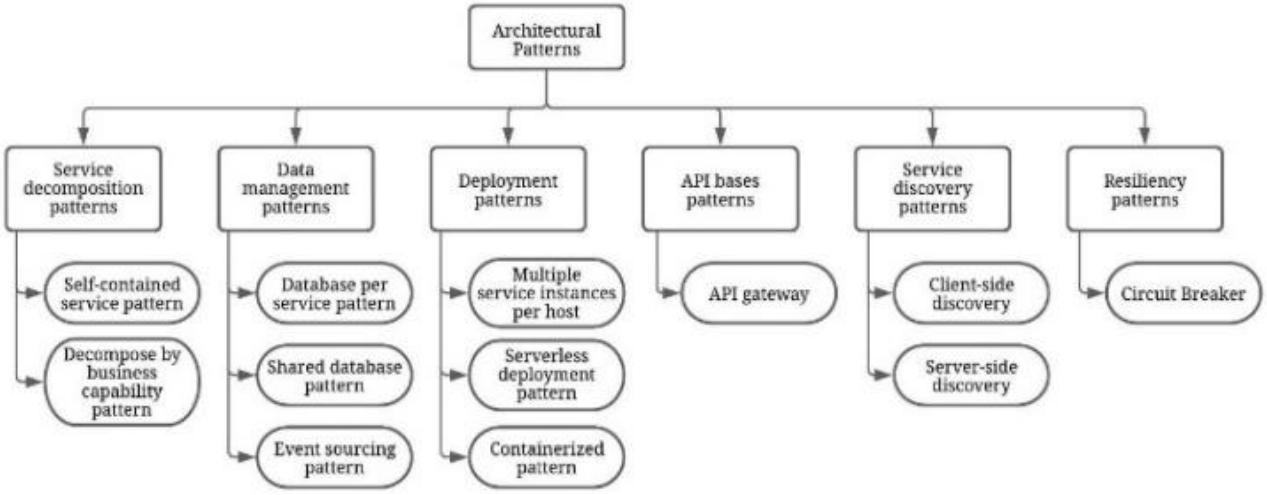

Fig. 4 Architectural patterns in a microservice architecture

Based on the above table 1, Spring Boot is the most popular and the number one trending microservice framework of the world, that comprises the most critical quality attributes. Go-micro-framework is also turning out to be famous in the industry because of its performance. Vert. X microservice framework does not give proper flexibility to bring the software quality attributes as required [63]. It is a slowly evolving framework compared to the Spring boot. But, Vert. X is implemented using the multi reactor pattern and therefore possesses good performance. The molecular framework is not widely popular in the microservice development industry. However, according to its architecture, it is suitable for high-performance microservices. World-famous multimillionaire companies have transited their systems to the microservice-based architecture to get more advantages in terms of the product, cost, and the process. Services in the microservice architecture develop, deploy and execute independently. Thus service needs proper technology to perform efficiently. Architectural patterns are more important to gain quality attributes of the system. If correct architectural patterns are not followed according to the software domain, will fail to

deliver high-quality software services to the client.
When considering the architectural pattern for the microservices, one needs to consider the following factors.

- Software solution understandability
- Modifiability
- Resiliency
- Performance
- Continuous deployment of the software solution
- Scalability
- Emerging technology adaptation
- Independency

In the software industry, several architectural patterns can be used to ensure the above-mentioned factors. Segregated the patterns into several areas to achieve the above factors (figure 4). There are various architecture patterns-based quality attributes in the systems [64]. Service decomposition patterns illustrate how the system can be independent of several services. Monolithic system holders can use this pattern to segregate their services to maximize the application performance. To use this pattern, architects need to have proper knowledge of the business domain, as well as the technology associated with the domain. The method of interacting with the data is critical to the system. There are several microservice architecture patterns defined to design the data management. Before designing the data management pattern, you need to study the scenarios of scaling the microservices, the requirement of different data sources such as RDBMS, NoSQL, data retentions, and the data flow. Based on the business requirement, architects can choose the data management patterns. Many studies show that event sourcing patterns are not good for transactional management business scenarios [65]. When scrutinizing modern trends of microservices, containerized deployments are more popular in the world. Certain people opt for server-less kind of solutions for less computational services. Cloud providers are also inventing server-less platform features as well as containerization supports. The most famous architectural pattern is the API-based pattern. Because most of the microservice expose through the API manager to governance the APIs which are exposed through the microservices. New API management platforms are invented for the software industry daily. People move to microservice architecture to attain better scalability in their services. Microservices can be scaled into many services based on the traffic load, and if the traffic is low, such a service can be brought down into the optimal value on demand. It is crucial to make configuration changes in and out of each scale. In such a scenario, can use service discovery patterns. Most software firms are currently concerned about software resilience. Resilience pattern is responsible for preventing the cascading failure to other services from one service network or service failure. The renowned pattern for resilience is the circuit breaker pattern.

C. What are the Main Motivations to Convert the Monolithic Application to Microservice Architecture?

Publication	References
Journal	[66], [67], [68], [69], [70]
Conference	[71], [72], [73], [74], [75], [76], [77]

Many people move to microservices to gain more advantages for their business and software. But, microservices are not the perfect match for every software domain. Some architects, developers, support teams, and other stakeholders have pains in using microservice architecture. Technology has gradually improved to resolve these challenges; yet, some of the pains could not be mitigated by the current technological advancements or solutions. This paper elaborates the findings on the challenge when using microservice architecture. Segregate the challenges into several phases such as design level challenges, implementation level challenges, and support level challenges.

There are two main types of microservice design. The first one is designing the microservice from the new business requirement and new solutions. Another type is converting monolithic systems or service-oriented traditional systems to microservice architecture. The first challenge in developing a microservice is to determine the scope of one service in a microservice architecture. The scope can vary from business domain to domain, software type, functional and non-functional requirements. Yet, a concise theory is not brought forward for the separation of microservices. Some researchers demonstrate that service should only need focus on the specific business flow; while another group of researchers suggests that service scope should be bound to one application domain. In the software industry, the scope of microservice is segregated into programming operational service on the overall microservice architecture such as DB service, messaging service, file read/write service, etc. Fast-growing and well-established frameworks like Spring boot are also built to support the industry practice. Hence, a clear principle to define the microservice scope is still ambiguous. Amidst the absence of a proper scope definition, it is challenging to contemplate the size of the microservice. Hence, based on that, can't determine the connection points within microservice architecture.

At the implementation level, the developer needs to detect the quality attribute. Services in the microservice architecture are deployed in the distributed environment. This could be different networks, multi-cloud or hybrid clouds. Therefore, data needs to be transferred to each service to complete the business requirement, which ultimately leads to vulnerabilities in the entire software solution. Data security across the microservice is a big concern in the microservice architecture. It is a complicated

assignment to preserve the integrity, confidentiality, and privacy of business data. Some frameworks support validating the security tokens such as JSON Web Token. But, validating the JWT token on each service will lead to an overall performance issue [78].

Another main concern behind the microservice is the application performance in terms of latency [79]. There are a lot of protocols, for instance, HTTP/HTTPS, gRPC, JMS, AMQP, and web sockets to service-to-service communication. Some protocols generate more complexities to the microservice architecture, while some do not support the cloud-native environments. Based on the domain of the business, service-to-service communication can be synchronous or asynchronous. Most of the developers use the HTTP/HTTPS protocol for service-to-service communication. Because of the latency issues, architects do not try to define the services accurately in the microservice architecture.

Nowadays, several languages are present to write the deployment scripts such as Ansible for server deployments and configurations, puppet and the helm chart for the Kubernetes deployment, and terraform for cloud-level deployments. In the microservice architecture, one single software solution has several services and needs to write deployment scripts to every microservice. Software deployment will take considerable time compared to traditional software systems. Introducing the CI/CD pipeline can reduce the time of deployment, but it will take additional effort and cost.

Supporting the microservice architecture is very challenging because of the service distribution. If microservices don't have the request tracing mechanism implemented, then support engineering will be facing a nightmare. They would have to reach each microservice and check on the root cause for the issue. Bringing the fault tolerance to the microservice architecture is crucial as many services are involved with the distributed environment. Each path needs to test for fail resilience and should ensure fault tolerance. The production system would need to have very strong monitoring. In a microservice architecture, enabling monitoring needs more effort and resources due to the small number of services deployed in the distributed environment.

V. TAXONOMY CLARIFICATION ON MICROSERVICES TECHNOLOGY TRENDS

Researchers have chosen 3 fundamental stages in the software lifecycle to construct the taxonomy [80] development, deployment, and operational (figure 5). After a critical review, identified the three main categories that microservice research is moving towards in near future AI, cloud, and architecture.

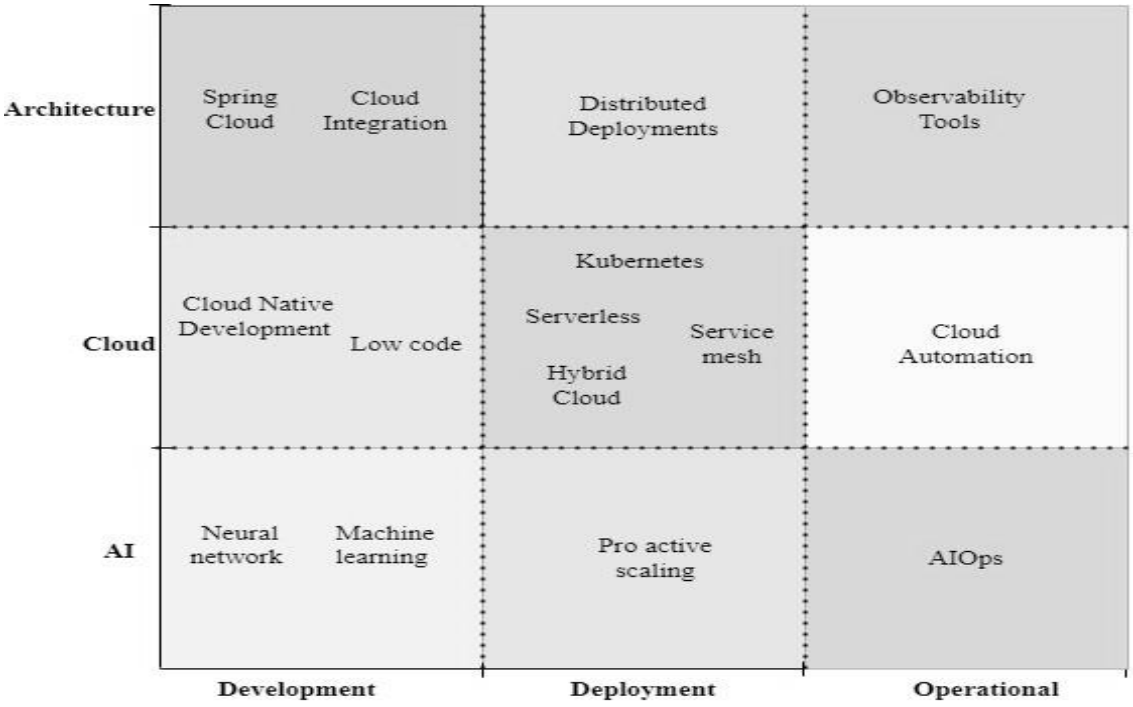

Fig. 5 Taxonomy on microservices future trends

Most of the monolithic systems are moved to microservice-based architecture to bring quality attributes to the system. Nowadays chief technology offices around the world's KPI is to bring the business software into the cloud-native platform. When moving to the cloud-native architecture, cost and application performance are the biggest

limitations. To get better performance with low cost in the cloud-native environment, the application should possess the capability to run on low computational specifications and need to perform with better results. Most of the microservice frameworks are now built according to the cloud-native architecture concepts. The review found that the most famous JAVA Spring framework massively supports cloud-native development. As an example, spring-cloud supports the GCP / Azure / AWS / Alibaba cloud Integration for microservices. Go micro kind of microservice frameworks enable high performance with the lightweight applications to achieve low cost. Other frameworks are actively working on developing cloud-native support libraries. The future of the microservice is mainly based on the cloud-native environment.

Most of the microservice-related research mainly focuses on the cloud-native concepts and integration with artificial intelligence to the microservice architecture. By critically evaluating the ongoing research, found out that most of the machine learning and neural network technologies are used for the predictive analysis in proactive scaling. But those kinds of implementations are needed for high-end computational resources. Or else, it is mandatory to use the cloud services (which are inclusive of certain costs). AIOps concepts mainly focus on microservices [81].
Most of the cloud providers invent serverless architecture services and function as a service. Cloud consumers are moving towards serverless because it provides on-demand computing resources. Also, it is very beneficial for the industry cost-wise. Researchers have introduced the architectural pattern for serverless architecture with several reference architectures for some domains. The entire application stack will be decentralized in the function as a service concept. Intelligence-driven microservices with the in-memory resources that conduct real-time analytics will be the trend in this concept. On the other hand, people try to deploy their solutions in hybrid cloud environments. The reason behind moving to hybrid cloud deployment is to minimize network latency based on geographical areas and to gain several cloud vendor services.

The industry is rapidly moving to microservice architecture to associate with considerable challenges as well. The main challenge is the performance in terms of latency because of the inter-service communication between microservice in the distributed environment. Skill for development of the microservices is another channel in the software industry. When the application is deployed into the production operation, the troubleshooting with the tracing is quite complicated in the microservice architecture. But currently, some of the tools are being developed to sort out challenges like Zipkin and Jager [82].

V. CONCLUSION AND FUTURE WORK

This systematic review extensively discusses the microservices using the three research questions by providing supportive data. Used the PRISMA model to conduct this systematic review. Research questions are chosen to capture all aspects of the microservice research area. Most people are moving their monolithic system to microservice architecture to achieve the quality attributes such as scalability, performance, security, maintainability, etc. In the current context, with the emergence of world pandemic situations, most of the services are served via online platforms and a lot of users are moving towards the online platforms to get the services.

Most software companies tend to develop scalable microservice to cater to a considerable workload. To achieve complete benefits of the cloud services application, developers need to change the application architecture to the cloud-native microservice architecture. The review found that most of the software frameworks are now supporting microservice development. Researchers introduce modern architectural patterns for microservice development in terms of integration patterns, data management, service segregation, traffic routing, and deployment patterns. Since microservice technology is a mature concept, most researchers now focus on developing the framework, patterns, and new technology advancement of the microservice concept. Software architecture-related future research tends to focus on the microservice integration patterns on distributed cloud environments. With the technological advancement and the architectural patterns, some of the challenges are overcome but they are still faced with several issues in the microservice architecture. The main concern of the microservice architecture is the latency because of the distributed services. Another problem is the troubleshooting issue on the microservice, which is challenging for the support engineers. Continuous research of this to find a solution for the performance issue in the microservice architecture. With the selected research questions, can comprehensively go through the microservice concepts and their related research areas.

The review identified the critical areas that the future of the microservice-related research is focused on and developed the taxonomy based on that. The researchers who are interested in the AI field can focus on the microservice proactive scaling in the cloud and containerization level using AI technologies. AIOps is also a new trend in the microservice operational layer which is used to identify the possible faults using advanced technologies of AI. Containerization and serverless technologies adaptation is the most trending microservice research area for people who are focusing on cloud-related research. Soon, most of the systems may move towards cloud-based systems. Hence, applications need to be micro-level to achieve all the features of the cloud. Therefore, future research needs to focus on the cloud-native development of the microservice architecture

and should prioritize resolving the problems identified in this research. Enterprise-grade software has a proper observability stack when moving to the microservices architecture. New research areas are open to research to further improve the microservice observability in the distributed cloud environment. Can conclude that microservices are evolving in countless areas to cater to the current user trends.

REFERENCES

[1] L. O'Brien, P. Merson, and L. Bass, Quality Attributes for Service-Oriented Architectures, in International Workshop on Systems Development in SOA Environments (SDSOA'07: ICSE Workshops 2007), Minneapolis, MN, USA. (2007) 3–3. doi: 10.1109/SDSOA.2007.10.

[2] N. Alshuqayran, N. Ali, and R. Evans, A Systematic Mapping Study in Microservice Architecture, in 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), Macau, China. (2016) 44–51. doi: 10.1109/SOCA.2016.15.

[3] N. Kratzke, A Brief History of Cloud Application Architectures, Appl. Sci. 8(8) (2018) 1368. doi: 10.3390/app8081368.

[4] S. Hassan, R. Bahsoon, and R. Kazman, Microservice Transition and its Granularity Problem: A Systematic Mapping Study, Softw. Pract. Exp. 50(9) (2020) 1651–1681. doi: 10.1002/spe.2869.

[5] H. Vural, M. Koyuncu, and S. Guney, A Systematic Literature Review on Microservices, in Computational Science and its Applications – ICCSA. 10409 (2017).

[6] O. Gervasi, B. Murgante, S. Misra, G. Borruso, C. M. Torre, A. M. A. C. Rocha, D. Taniar, B. O. Apduhan, E. Stankova, and A. Cuzzocrea, Eds. Cham: Springer International Publishing. (2017) 203–217. doi: 10.1007/978-3-319-62407-5_14.

[7] James Lewis, Microservices, Martinfowler.com. [Online]. Available: https://martinfowler.com/articles/microservices.html

[8] C. Pahl and P. Jamshidi, Microservices: A Systematic Mapping Study: in Proceedings of the 6th International Conference on Cloud Computing and Services Science, Rome, Italy. (2016) 137–146. doi: 10.5220/0005785501370146.

[9] J. Soldani, D. A. Tamburri, and W.-J. Van Den Heuvel, The Pains and Gains of Microservices: A Systematic Grey Literature Review, J. Syst. Softw. 146 (2018) 215–232. doi: 10.1016/j.jss.2018.09.082.

[10] Vilnius Gediminas Technical University, Saultekio al. 11, LT-10223 Vilnius, O. Pozdniakova, D. Mažeika, and Vilnius Gediminas Technical University, Saultekio al. 11, LT-10223 Vilnius, Systematic Literature Review of the Cloud-ready Software Architecture, Balt. J. Mod. Comput. 5(1) (2017) 124–135. doi: 10.22364/bjmc.2017.5.1.08.

[11] J. Opara-Martins, R. Sahandi, and F. Tian, Critical Analysis of Vendor Lock-In and its Impact on Cloud Computing Migration: A Business Perspective, J. Cloud Comput. 5(1) (2016) 4. doi: 10.1186/s13677-016-0054-z.

[12] J. Ghofrani and D. Lübke, Challenges of Microservices Architecture: A Survey on the State of The Practice. 8.

[13] IEEE Xplore. [Online]. Available: www.ieeexplore.ieee.org/Xplore/home.jsp

[14] ACM Digital Library. [Online]. Available: www.dl.acm.org/

[15] CiteSeerX. [Online]. Available: www.citeseerx.ist.psu.edu/index;jsessionid=16DB95ECCF154FE7425AFA00F9934740

[16] ScienceDirect.com | Science, Health and Medical Journals, Full-Text Articles and Books. [Online]. Available: www.sciencedirect.com/

[17] M. J. Page et al., PRISMA 2020 Explanation and Elaboration: Updated Guidance and Exemplars for Reporting Systematic Reviews, BMJ. (2021) 160. doi: 10.1136/bmj.n160.

[18] A. Bucchiarone, N. Dragoni, S. Dustdar, S. T. Larsen, and M. Mazzara, From Monolithic to Microservices: An Experience Report from the Banking Domain, IEEE Softw. 35(3) (2018) 50–55. doi: 10.1109/MS.2018.2141026.

[19] V. Lenarduzzi, F. Lomio, N. Saarimäki, and D. Taibi, Does Migrating a Monolithic System to Microservices Decrease the Technical Debt? J. Syst. Softw. 169 (2020) 110710. doi: 10.1016/j.jss.2020.110710.

[20] F. Auer, V. Lenarduzzi, M. Felderer, and D. Taibi, From Monolithic Systems to Microservices: An Assessment Framework, Inf. Softw. Technol. 137 (2021) 106600. doi: 10.1016/j.infsof.2021.106600.

[21] O. Al-Debagy and P. Martinek, A Comparative Review of Microservices and Monolithic Architectures, in 2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI), Budapest, Hungary. (2018) 000149–00154. doi: 10.1109/CINTI.2018.8928192.

[22] M. Villamizar et al., Evaluating the Monolithic and the Microservice Architecture Pattern to Deploy Web Applications in the Cloud, In 2015 10th Computing Colombian Conference (10CCC), Bogota, Colombia. (2015) 583–590. doi: 10.1109/ColumbianCC.2015.7333476.

[23] 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, Japan. (2018) 571–580. doi: 10.1109/APSEC.2018.00072.

[24] G. Mazlami, J. Cito, and P. Leitner, Extraction of Microservices from Monolithic Software Architectures, in 2017 IEEE International Conference on Web Services (ICWS), Honolulu, HI, USA. (2017) 524–531. doi: 10.1109/ICWS.2017.61.

[25] 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, in 2020 IEEE International Conference on Software Architecture (ICSA), Salvador, Brazil. (2020) 157–168. doi: 10.1109/ICSA47634.2020.00023.

[26] J. Kazanavicius and D. Mazeika, Migrating Legacy Software to Microservices Architecture, in 2019 Open Conference of Electrical, Electronic and Information Sciences (eStream), Vilnius, Lithuania. (2019) 1–5. doi: 10.1109/eStream.2019.8732170.

[27] V. Velepucha and P. Flores, Monoliths to microservices - Migration Problems and Challenges: A SMS, in 2021 Second International Conference on Information Systems and Software Technologies (ICI2ST), Quito, Ecuador. (2021) 135–142. doi: 10.1109/ICI2ST51859.2021.00027.

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

[29] S. Sarkar, G. Vashi, and P. P. Abdulla, Towards Transforming an Industrial Automation System from Monolithic to Microservices, in 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), Turin. (2018) 1256–1259. doi: 10.1109/ETFA.2018.8502567.

[30] 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, Porto Portugal. (2018) 1–6. doi: 10.1145/3234152.3234195.

[31] Z. Ren et al., Migrating Web Applications from Monolithic Structure to Microservices Architecture, in Proceedings of the Tenth Asia-Pacific Symposium on Internetware, Beijing China. (2018) 1–10. doi: 10.1145/3275219.3275230.

[32] A. K. Kalia et al., Mono2Micro: An AI-based Toolchain for Evolving Monolithic Enterprise Applications to a Microservice Architecture, in Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event USA. (2020) 1606–1610. doi: 10.1145/3368089.3417933.

[33] S. Weerasinghe and I. Perera, An Exploratory Evaluation of Replacing ESB with Microservices in Service-Oriented Architecture, Presented at the International Research Conference on Smart Computing and Systems Engineering. (2021).

[34] T. Ueda, T. Nakaike, and M. Ohara, Workload characterization for microservices, in 2016 IEEE International Symposium on Workload Characterization (IISWC). (2016) 1–10. doi: 10.1109/IISWC.2016.7581269.
[35] A. de Camargo, I. Salvadori, R. dos S. Mello, and F. Siqueira, An Architecture to Automate Performance Tests on Microservices, in Proceedings of the 18th International Conference on Information Integration and Web-Based Applications and Services, New York, NY, USA. (2016) 422–429. doi: 10.1145/3011141.3011179.
[36] M. Gribaudo, M. Iacono, and D. Manini, Performance Evaluation of Replication Policies in Microservice Based Architectures, Electron. Notes Theor. Comput. Sci. 337 (2018) 45–65. doi: 10.1016/j.entcs.2018.03.033.
[37] A. Balalaie, A. Heydarnoori, and P. Jamshidi, Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture, IEEE Softw. 33(3) (2016) 42–52. doi: 10.1109/MS.2016.64.
[38] G. Marquez, F. Osses, and H. Astudillo, Review of Architectural Patterns and Tactics for Microservices in Academic and Industrial Literature, IEEE Lat. Am. Trans. 16(9) (2018) 2321–2327. doi: 10.1109/TLA.2018.8789551.
[39] H. Suryotrisongko, D. P. Jayanto, and A. Tjahyanto, Design and Development of Backend Application for Public Complaint Systems Using Microservice Spring Boot, Procedia Comput. Sci. 124 (2017) 736–743. doi: 10.1016/j.procs.2017.12.212.
[40] H. Dinh-Tuan and M. Mora-Martinez, Development Frameworks for Microservice-Based Applications: Evaluation and Comparison. 9.
[41] T. de Oliveira Rosa, J. F. L. Daniel, E. M. Guerra, and A. Goldman, A Method for Architectural Trade-Off Analysis Based on Patterns: Evaluating Microservices Structural Attributes, in Proceedings of the European Conference on Pattern Languages of Programs 2020, Virtual Event Germany. (2020) 1–8. doi: 10.1145/3424771.3424809.
[42] J. Dobaj, J. Iber, M. Krisper, and C. Kreiner, A Microservice Architecture for the Industrial Internet-of-Things, in Proceedings of the 23rd European Conference on Pattern Languages of Programs, Irsee Germany. (2018) 1–15. doi: 10.1145/3282308.3282320.
[43] G. Marquez and H. Astudillo, Actual Use of Architectural Patterns in Microservices-Based Open Source Projects, in 2018 25th Asia-Pacific Software Engineering Conference (APSEC), Nara, Japan. (2018) 31–40. doi: 10.1109/APSEC.2018.00017.
[44] M. K and M. P, Evaluation of Data Storage Patterns in Microservices Architecture, in 2020 IEEE 15th International Conference of System of Systems Engineering (SoSE), Budapest, Hungary. (2020) 373–380. doi: 10.1109/SoSE50414.2020.9130516.
[45] F. Osses, G. Márquez, and H. Astudillo, Exploration of Academic and Industrial Evidence about Architectural Tactics and Patterns in Microservices, in Proceedings of the 40th International Conference on Software Engineering: Companion Proceedings, Gothenburg Sweden. (2018) 256–257. doi: 10.1145/3183440.3194958.
[46] F. Montesi and J. Weber, From the Decorator Pattern to Circuit Breakers in Microservices, in Proceedings of the 33rd Annual ACM Symposium on Applied Computing, Pau France. (2018) 1733–1735. doi: 10.1145/3167132.3167427.
[47] H. Harms, C. Rogowski, and L. Lo Iacono, Guidelines for Adopting Frontend Architectures and Patterns in Microservices-Based Systems, in Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, Paderborn Germany. (2017) 902–907. doi: 10.1145/3106237.3117775.
[48] G. Márquez and H. Astudillo, Identifying Availability Tactics to Support Security Architectural Design of Microservice-Based Systems, in Proceedings of the 13th European Conference on Software Architecture - ECSA '19, Paris, France. 2 (2019) 123–129. doi: 10.1145/3344948.3344996.
[49] F. Li et al., Microservice Patterns for the Life Cycle of Industrial Edge Software, in Proceedings of the 23rd European Conference on Pattern Languages of Programs, Irsee Germany. (2018) 1–11. doi: 10.1145/3282308.3282313.
[50] 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), Nakhonsithammarat, Thailand. (2017) 1–4. doi: 10.1109/JCSSE.2017.8025912.
[51] M. Tusjunt and W. Vatanawood, Refactoring Orchestrated Web Services into Microservices using Decomposition Pattern, in 2018 IEEE 4th International Conference on Computer and Communications (ICCC), Chengdu, China. (2018) 609–613. doi: 10.1109/CompComm.2018.8781036.
[52] A. Akbulut and H. G. Perros, Software Versioning with Microservices through the API Gateway Design Pattern, in 2019 9th International Conference on Advanced Computer Information Technologies (ACIT), Ceske Budejovice, Czech Republic. (2019) 289–292. doi: 10.1109/ACITT.2019.8779952.
[53] S. Vergara, L. Gonzalez, and R. Ruggia, Towards Formalizing Microservices Architectural Patterns with Event-B, in 2020 IEEE International Conference on Software Architecture Companion (ICSA-C), Salvador, Brazil. (2020) 71–74. doi: 10.1109/ICSA-C50368.2020.00022.
[54] S. du Plessis, B. Mendes, and N. Correia, A Comparative Study of Microservices Frameworks in IoT Deployments, in 2021 International Young Engineers Forum (YEF-ECE), Caparica / Lisboa, Portugal. (2021) 86–91. doi: 10.1109/YEF-ECE52297.2021.9505049.
[55] D. Lu, D. Huang, A. Walenstein, and D. Medhi, A Secure Microservice Framework for IoT, in 2017 IEEE Symposium on Service-Oriented System Engineering (SOSE), San Francisco, CA, USA. (2017) 9–18. doi: 10.1109/SOSE.2017.27.
[56] X. Liu, S. Jiang, X. Zhao, and Y. Jin, A Shortest-Response-Time Assured Microservices Selection Framework, in 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC), Guangzhou. (2017) 1266–1268. doi: 10.1109/ISPA/IUCC.2017.00192.
[57] R. Manciola Meloca, R. Ré, and A. Luis Schwerz, An Analysis of Frameworks for Microservices, in 2018 XLIV Latin American Computer Conference (CLEI). (2018) 542–551. doi: 10.1109/CLEI.2018.00071.
[58] Y. Wang, L. Cheng, and X. Sun, Design and Research of Microservice Application Automation Testing Framework, in 2019 International Conference on Information Technology and Computer Application (ITCA), Guangzhou, China. (2019) 257–260. doi: 10.1109/ITCA49981.2019.00063.
[59] L. Liu, X. He, Z. Tu, and Z. Wang, MV4MS: A Spring Cloud-Based Framework for the Co-Deployment of Multi-Version Microservices, in 2020 IEEE International Conference on Services Computing (SCC), Beijing, China. (2020) 194–201. doi: 10.1109/SCC49832.2020.00033.
[60] Spring Boot. [Online]. Available: www.spring.io/projects/spring-boot
[61] (2021). A. Aslam, Go Micro. Accessed. [Online]. Available: https://github.com/asim/go-micro
[62] Molecular - Progressive Microservices Framework for Node.js. [Online]. Available: www.Moleculer.Services/Index.Html
[63] (2021). Eclipse Vert.x. [Online]. Available: https://vertx.io/
[64] S. Brenner, T. Hundt, G. Mazzeo, and R. Kapitza, Secure Cloud Micro Services Using Intel SGX. (2017) 177–191. doi: 10.1007/978-3-319-59665-5_13.
[65] Chris Richardson, Microservices Patterns. Manning Publications. (2018).
[66] (2021). A Whole System Based on Event Sourcing is an Anti-Pattern. InfoQ. [Online]. Available: www.infoq.com/news/2016/04/event-sourcing-anti-pattern/.
[67] C. Esposito, A. Castiglione, and K.-K. R. Choo, Challenges in Delivering Software in the Cloud as Microservices, IEEE Cloud Comput. 3(5) (2016) 10–14. doi: 10.1109/MCC.2016.105.
[68] F. Rademacher, J. Sorgalla, and S. Sachweh, Challenges of Domain-Driven Microservice Design: A Model-Driven Perspective, IEEE Softw. 35(3) (2018) 36–43. doi: 10.1109/MS.2018.2141028.

[69] B. Götz, D. Schel, D. Bauer, C. Henkel, P. Einberger, and T. Bauernhansl, Challenges of Production Microservices, Procedia CIRP. 67 (2018) 167–172. doi: 10.1016/j.procir.2017.12.194.

[70] N. C. Mendonca, P. Jamshidi, D. Garlan, and C. Pahl, Developing Self-Adaptive Microservice Systems: Challenges and Directions, IEEE Softw. 38(2) (2021) 70–79. doi: 10.1109/MS.2019.2955937.

[71] T. Cerny et al., On Code Analysis Opportunities and Challenges for Enterprise Systems and Microservices, IEEE Access. 8 (2020) 159449–159470. doi: 10.1109/ACCESS.2020.3019985.

[72] M. Kleehaus and F. Matthes, Challenges in Documenting Microservice-Based IT Landscape: A Survey from an Enterprise Architecture Management Perspective, in 2019 IEEE 23rd International Enterprise Distributed Object Computing Conference (EDOC), Paris, France. (2019) 11–20. doi: 10.1109/EDOC.2019.00012.

[73] 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), Rawalpindi, Pakistan. (2019) 82–87. doi: 10.1109/COMTECH.2019.8737831.

[74] J. Fritzsch, J. Bogner, S. Wagner, and A. Zimmermann, Microservices Migration in Industry: Intentions, Strategies, and Challenges, in 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), Cleveland, OH, USA. (2019) 481–490. doi: 10.1109/ICSME.2019.00081.

[75] S. Eismann, C.-P. Bezemer, W. Shang, D. Okanović, and A. van Hoorn, Microservices: A Performance Tester's Dream or Nightmare?, in Proceedings of the ACM/SPEC International Conference on Performance Engineering, Edmonton AB Canada. (2020) 138–149. doi: 10.1145/3358960.3379124.

[76] T. Yarygina and A. H. Bagge, Overcoming Security Challenges in Microservice Architectures, in 2018 IEEE Symposium on Service-Oriented System Engineering (SOSE), Bamberg. (2018) 11–20. doi: 10.1109/SOSE.2018.00011.

[77] I. Pigazzini, F. A. Fontana, V. Lenarduzzi, and D. Taibi, Towards Microservice Smell Detection, in Proceedings of the 3rd International Conference on Technical Debt, Seoul Republic of Korea. (2020) 92–97. doi: 10.1145/3387906.3388625.

[78] W. K. G. Assunção, J. Krüger, and W. D. F. Mendonça, Variability Management Meet Microservices: Six Challenges of Re-Engineering Microservice-Based Webshops, in Proceedings of the 24th ACM Conference on Systems and Software Product Line: Volume A - Volume A, Montreal Quebec Canada. (2020) 1–6. doi: 10.1145/3382025.3414942.

[79] Y. Sun, S. Nanda, and T. Jaeger, Security-as-a-Service for Microservices-Based Cloud Applications, in 2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom). (2015) 50–57. doi: 10.1109/CloudCom.2015.93.

[80] N. Kratzke and P.-C. Quint, Investigation of Impacts on Network Performance in the Advance of a Microservice Design, in Cloud Computing and Services Science, Cham. (2017) 187–208. doi: 10.1007/978-3-319-62594-2_10.

[81] T. Saravanan, S. Jha, G. Sabharwal, and S. Narayan, Comparative Analysis of Software Life Cycle Models, in 2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN). (2020) 906–909. doi: 10.1109/ICACCCN51052.2020.9362931.

[82] Y. Dang, Q. Lin, and P. Huang, AIOps: Real-World Challenges and Research Innovations, in 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), Montreal, QC, Canada. (2019) 4–5. doi: 10.1109/ICSE-Companion.2019.00023.

[83] M. S D and D. M., Distributed Request Tracing using Zipkin and Spring Boot Sleuth, Int. J. Comput. Appl. 175 (2020) 35–37. doi: 10.5120/ijca2020920617.