

# Analysis of Monolithic and Microservices System Architectures for an E-Commerce Web Application

Dr. Meenakshi Thalor<sup>1\*</sup>, Shivprasad R. Allur<sup>2</sup>, Dr. Veena Suhas Bhende<sup>3</sup>, Dr. Amrapali Chavan<sup>4</sup>

Submitted: 12/03/2024    Revised: 27/04/2024    Accepted: 04/05/2024

**Abstract:** Selection of a system architecture for deployment of a web application is always a crucial decision as lots of planning and implementation involves. As monolithic architecture offers a simpler and more straightforward approach and better results for a small-scale approach whereas microservices architecture would be an expensive approach. In addition, monolithic architecture gives satisfactory results compared to microservices architecture while having low traffic. However, the error percentage of monolithic architecture is extremely high while having heavy traffic whereas microservices architecture handles heavy traffic with a very low error percentage. This research paper explores the relative merits of monolithic and microservices architecture for E-Commerce web applications, using Express JS and Node JS as the primary technologies. The study provides a comprehensive examination of the two architecture patterns and employs a practical approach to demonstrate the differences. The architecture is compared based on metrics such as latency, throughput, response-time, error percentage and cost. The findings indicate that when it comes to large and complex applications, microservices architecture outperforms monolithic architecture in terms of scalability and reliability. The appropriate choice of architecture pattern should be determined by the unique needs of the project. The objective of this research is to evaluate the monolithic and microservices architectures for an ecommerce use case, and to propose guidelines for small- and large-scale enterprises on which architecture to implement.

**Keywords:** Microservices, Monolithic, Software Architecture, Comparison, Architectural Metrics, Performance Testing

## 1. Introduction

Modern businesses have used microservices architecture, including Amazon, Net-flix, Uber, and Spotify. Financial, e-commerce, and travel service providers are switching from monolithic to microservices architecture.

The term "monolithic architecture" refers to a single-tiered software application in which various components are combined into a single programme from a single platform. Despite the fact that the application contains multiple components/ modules/services, it is built and deployed as a single application for all platforms (desktop, mobile and tablet). Advantages of monolithic architecture include:

- Easy to develop and deploy
- Simple to understand and maintain
- Good for small-scale applications
- Cost-effective compared to microservices architecture

However, monolithic architecture can become unwieldy and difficult to maintain as applications grow in size and complexity, leading to a greater risk of codebase degradation and long development cycles. This is one of the reasons why microservices architecture has gained popularity.

Microservices architecture is a software design approach

where an application is composed of small, independently deployable services. Each service is focused on performing a specific business function and can be developed, deployed, and scaled independently of the other services. Advantages of microservices architecture include:

- Improved scalability
- Better fault tolerance
- Enhanced maintainability
- Increased deployment velocity
- Facilitation of a polyglot development environment
- Ability to evolve systems incrementally
- Better support for continuous delivery and deployment
- Easier integration with diverse systems and technologies

However, microservices architecture can be more complex to set up and manage compared to monolithic architecture and can also lead to increased operational overhead and communication overhead between services. It also requires a high level of technical expertise.

The paper compares the monolithic and microservices architecture specifically for E-Commerce web applications. Based on the research conducted, the architectures are compared on the metrics such as latency, throughput, error percentage, response-time and cost. Because of the ever-increasing popularity of media sites such as Instagram, YouTube, Facebook, etc. e-commerce businesses are also gaining popularity as it has become easier to market their

<sup>1,2,3,4</sup> AISSMS Institute of Information Technology, Pune, Maharashtra, India

\* Corresponding Author Email: meenakshi.thalor@aiissmsioit.org

products on a huge scale. With this increasing popularity, consequently, network traffic is bound to increase. This can result in various problems for small businesses such as low tolerance and inability to scale. Whereas, small scale businesses should not adopt microservices architecture as it would be expensive and the utilization would be insufficient, which in turn would make the business harder to sustain. Hence, the research paper focuses on selecting the appropriate architecture for an e-commerce web application based on the necessities, requirements and future scope. This paper does not focus on any particular domain, platform, or functionality that may be unique to a certain application or service. Instead, we are using a common and generic scenario that can be applied to any application or service that follows the monolithic or microservices architecture. This allows us to compare the architectures in a fair and objective way, without introducing any bias or confounding factors that may affect the results.

## 2. Related Work

The authors of the work by Raj V. et al. [1] offered a comparison of a web application created utilizing both Service Oriented Architecture (SOA) and microservices architectures. Two separate parameters are used in the comparison: 1) Architectural metrics for complexity; 2) load testing for performance. The results demonstrated that even though the microservices design is sophisticated, it responds to HTTP re-quests far more quickly than SOA services.

Grzegorz Blinowski et al. [2], compares the Monolithic and Microservice Architecture. Microservices-based architecture has gained widespread popularity due to its advantages, such as improved availability, fault tolerance, and horizontal scalability, as well as greater software development agility. The key lesson is on a single machine, a monolithic performs better than its microservice based counterpart.

Microservices and monolithic architectures are compared in terms of performance in Omar Al-study Debagy's [3], in order to ascertain how these architectures perform in various scenarios using various testing setups. This paper comes to the conclusion that monolithic applications and microservices can perform similarly when the application is under normal load. A monolithic application may perform marginally better than a microservices application under a light load of fewer than 100 users.

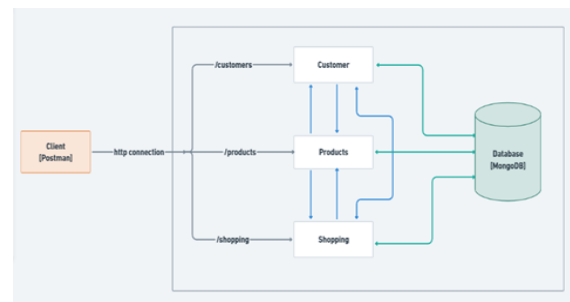
The authors of the work by Konrad Gos et al.[4] compared the Monolithic and Microservices architecture using the Gatling load testing tool. The tests were per-formed on PC with Ubuntu 18.04.2 LTS operating system. The applications were deployed with Docker. This paper compares the monolithic and microservices architecture on different parameters like Architecture performance,

response time by sending a number of HTTP GET and POST requests.

From the above research it is clear that processing requests using a microservices architecture is quicker than using a service-oriented design and monolithic architecture. However, not every situation calls for it. For simple, light-weight systems that don't need to support many concurrent users, monolithic design is preferable. Mono-lithic architecture outperformed microservices design in concurrency testing by 6% in terms of throughput. The microservice architecture is more effective when the programme has to handle more requests, according to load testing. There are benefits and drawbacks to both monolithic and microservices systems, with monolithic architecture being more effective, having less overhead, and being simple to extend.

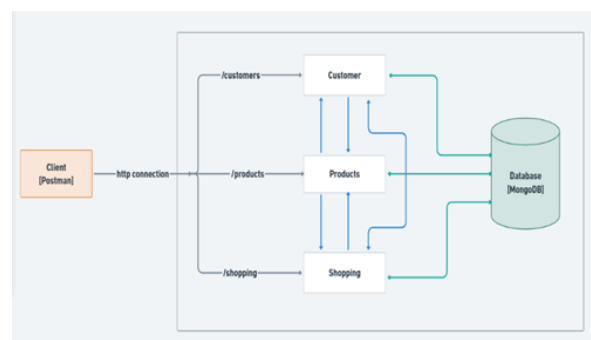
## 3. Methodology

To begin with the experiment, the development of the e-commerce web application using the monolithic architecture was designed as shown in fig.1.



**Fig. 1: Monolithic Architecture**

Designed website consist of three models: Customers, Products and Shopping. The Customers model contains the schema of the customers, Products holds the schema of the products and Shopping holds the schema of orders and the status of the orders. All the three models store their data in the same database as various collections. After successful development, testing of the routes and database is done using Post-man by sending HTTP Requests to the various routes. After developing the mono-lithic architecture, web application is migrated to microservices architecture as shown in fig. 2.



**Fig. 2: Microservices Architecture**

The three models are converted into independent services, i.e. the Customer Service, Products Service and Shopping Service. Each service has its own independent database to which it is connected. One of the features of microservices architecture is the ability to communicate with each other while also being independent, this was achieved by using RabbitMQ.

#### 4. Implementation

To compare the two architectures, two similar web applications developed using the same technologies. Different technologies like NodeJS and ExpressJS used for web application development, MongoDB Atlas and NoSQL cloud for database, Postman for load testing and Apache JMeter is used for generating experimental results.

The rationale for choosing this technology stack was based on the factors such as speed, efficiency, scalability and flexibility. Additionally, the compatibility and supportability of this technology stack with other tools used in the experiment were considered.

Node.js is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code on the server-side. It allows developers to build fast, scalable, and efficient server-side applications using JavaScript.

Express.js is a popular Node.js framework that makes it easier to build server-side web applications. It provides a robust set of features for web and mobile applications and simplifies the process of building RESTful APIs and web applications. Furthermore, it provides features like routing, middleware, templates, and more, and makes it easier for developers to build scalable, robust, and maintainable server-side applications.

MongoDB Atlas is a fully managed cloud database service developed by MongoDB Inc. It provides a document-based database system with a flexible schema that can be easily integrated into modern applications. It offers various features like automatic scaling, multi-cloud deployment, 24/7 support, and backup and recovery. With MongoDB Atlas, users can host their databases on cloud platforms such as Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure.

Postman is a popular API development and testing tool. It provides a platform for API developers to send, test, and document API requests and responses. Postman allows users to design and test APIs with ease, enabling efficient and seamless collaboration across teams.

Apache JMeter is a free and open-source load testing tool that can be used to measure the performance of various services, including web applications, databases, and APIs. It allows users to simulate a heavy load on a server, network, or object to test its strength or to analyze overall performance under different load types. JMeter can generate

many concurrent users and send requests to the target system. The results can be analyzed and visualized to identify performance bottlenecks and provide recommendations for improvement.

RabbitMQ is open-source message broker software that implements the Advanced Message Queuing Protocol (AMQP). It allows for decoupled communication between applications by storing messages in a queue and allowing consuming applications to process them as they become available. RabbitMQ can handle high volume and high throughput of messages and can support multiple messaging patterns including publish/subscribe, request/reply, and message routing.

All the services are connected to the Message Queue by the publish/subscribe messaging pattern. Each service has its own queue and listens to another queue. When one service publishes its payload, the payload contains a binding key, according to the binding key the exchanger sends the payload to the specific queue where the targeted service is listening. The target service executes an event depending on the payload received. Also if the targeted service goes offline for any reason, as soon as it comes back online, it receives the all payloads from the Message Queue and quickly synchronizes itself with the other services. In this way, all the independent services can communicate with each other.

As shown in the Fig. 2, the microservices architecture requires an API Gateway. The API Gateway forwards the HTTP Requests to

selected from each group of values. Pooling fastens the training process and provide focus on vital features of image.

A MLP come in picture at fully connected layer. MLP takes 1-d vector which is output of previous layer, and it provides list of probabilities of different possible classes. The class which gets highest probability is considered as final classification of input image.

In proposed work, a robust and efficient 5 layer one dimensional CNN model is trained on the Massachusetts Institute of Technology-Beth Israel Hospital (MIT-BIH) arrhythmia database. All trails of experiment are run for 500 epochs with learning rate in the range [0.01-0.5] using 32 filters, 5 kernel size and 1 strides value.

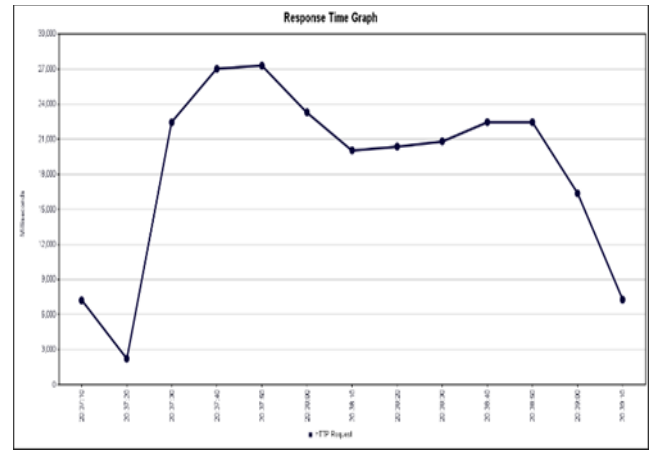
#### 5. Results and Discussion

For better understanding of the architectures, load testing experiments were conducted on various amounts of HTTP Request samples starting from low to high. Load Testing was performed on the system with following specifications i.e. Processor - 11th Gen Intel(R) Core (TM) i7 @ 3.40GHz, RAM - 16.0 GB

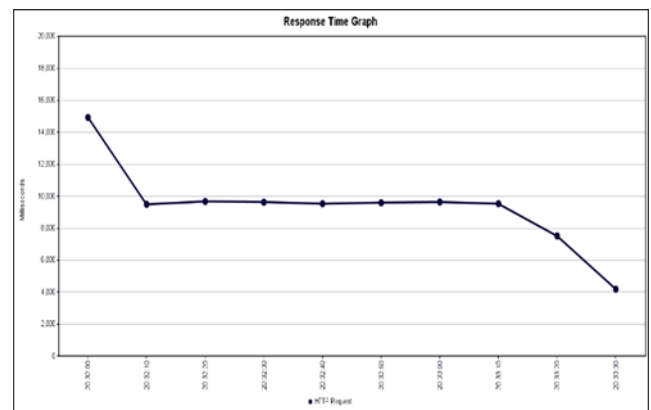
Commodity or standard computing resources in our test case

are selected, to mimic the application’s performance under practical industrial settings. This may not correspond to the precise outcomes, but it can offer significant insights for our analysis. The real outcomes after employing high-end computing resources could exhibit improved response time, reduced CPU utilization and enhanced results compared to our analysis.

Fig. 3 and fig. 4 show the response time graph of monolithic architecture and microservice architecture respectively. In the experiment, 1000 virtual users configure to transmit the HTTP requests. Response time is the time taken by the request to reach the server and get a response from the server. In the graph, the Y-axis represents the response time of each request in milliseconds and the X-axis represents the HTTP Requests.



**Fig. 3** Monolithic Response Time Graph with 10,000 samples



**Fig. 4** Microservice Response Time Graph with 10,000 samples

**Table 1:** Comparison Results

Label	HTTP Request (1st Run)		HTTP Request (2nd Run)		HTTP Request (3rd Run)	
# Samples	10000		20000		30000	
	Monolithic	Microservices	Monolithic	Microservices	Monolithic	Microservices
Average (ms)	11370	9137	19400	9247	22186	10877
Min (ms)	1	40	1	40	1	40
Max (ms)	39031	31982	56766	29226	51053	35790
Std. Dev.	11607.23	10897.00	10918.31	12458.36	8377.75	134223.80
Error %	15.03%	9.00%	51.20%	4.73%	62.01%	3.13%
Throughput	86.8/sec	108.4/sec	51.3/sec	107.7/sec	45.0/sec	91.4/sec
Received KB/sec	200.93	119.46	108.64	111.48	92.58	94.54
Sent KB/sec	5.63	12.13	5.81	12.62	5.79	10.89

Table 1 provides a summary report i.e. insightful data for the performance testing. This report displays average,

minimum and maximum response time taken by the requests in milliseconds. The “Std. Dev.” row represents the

deviation from the average value of response time. “Error %” represents the percentage of failed requests throughout the test duration. It is calculated as (No. of failed requests) / (Total No. of requests sent). The throughput row represents the number of requests processed by the server per second.

In the 1st run, it is clear that for 10,000 samples microservices provide better results and the monolithic architecture results to have average performance. The 2nd run has 20,000 samples. Compared to the previous run, the monolithic architecture began to get worse with an increasing number of samples and the microservices architecture gives approximately similar results. The 3rd run has 30,000 samples. The monolithic architecture continues to worsen, whereas the microservices architecture provides similar results showing its ability to handle high traffic.

Table 2 shows the average readings that are calculated and the results of the runs, which conclude that the monolithic architecture gives better results for lower numbers of samples and keeps deteriorating as the number of samples goes on increasing. However, the microservices architecture keeps on giving similar results for a high number of samples. This indicates that the microservices architecture is robust and can balance heavy load which is ideal for handling thousands of users simultaneously.

Table 2 Average Summary Report (Monolithic vs Microservices)

Label	Average Reading	
	Monolithic	Microservices
Average (ms)	17652	9753.66
Min (ms)	1	40
Max (ms)	48950	32332.66
Std. Dev.	10301.09	12268.72
Error %	42.74	5.62
Throughput	61.03	102.5

Fig. 9 graphs show strange emissions in the monolithic architecture, and it was speculated that the bottleneck in the monolithic architecture was caused by MongoDB rather than the monolithic application itself. However, after connecting all the services to the same cluster used by the monolithic architecture, the microservices response time graph remained unchanged. Table 3 shows that, irrespective of MongoDB clusters or multiple MongoDB clusters used in the microservices architecture, microservices architecture excels monolithic architecture.

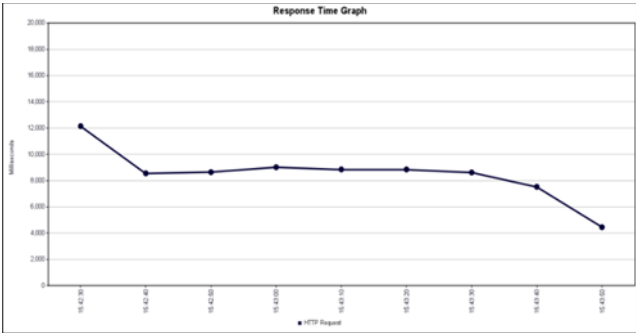


Fig. 9: Microservice Response Time Graph using Single MongoDB Instance

Table 3 Microservices Summary Report using Single MongoDB Instance

Label	HTTP Request (1st Run)
# Samples	10000
Average (ms)	8051
Min (ms)	28
Max (ms)	26552
Std. Dev.	10397.31
Error %	8.97%
Throughput	123.0/sec
Received KB/sec	253.84
Sent KB/sec	13.78

To conclude, a decision should be made according to the e-commerce business. The test cases are designed for generic ecommerce web applications and do not account for any specific scenario. Moreover, the analysis and test cases may differ depending on the computing resources and hardware used. The cases are as follows:

1. If you have a small business and have a network traffic of a few hundred users simultaneously, a backend server with monolithic architecture should suffice. This will help you to have low cost and maintenance for the e-commerce business and will also help your business sustain.
2. If you are a small business but are having or expecting to have high network traffic, you need an architecture which will help you scale easily and quickly. In that case, the microservices architecture will help you to scale your systems, allow you to make updates and also help your business to avoid loss of customers and will help your business. Combined with the services provided by various cloud platforms such as AWS, Microsoft Azure, Google Cloud Platform, your business will have a very robust and efficient e-commerce web application. If you are already

having an existing system, you might have to consider migrating to the microservices architecture.

3. If you are having big business and consider having a web application which is expected to have low traffic or developing a web application for your organization for a limited geographical region such as a country, you can develop a simple backend system having the monolithic architecture. This will be sufficient for your application, and you can also save money because of easy development, low system specification and low maintenance cost.

4. If you are having a big business and having heavy traffic on your e-commerce web application such as a worldwide business or developing a web application for your organization and the employees from all around the world are accessing it, you should develop a system having the microservices architecture. Considering this scenario, coupling the microservices architecture with Kuber-netes and other services provided by various cloud platforms such as AWS, Microsoft Azure, Google Cloud Platform, will be a great investment for the business.

## 6. Conclusion

Monolithic architecture and microservices architecture both have their pros and cons. In the research paper, the architecture, implementation, methodology, as well as the results and findings in detail. Based on these findings, we can say that the project stakeholder has to make an appropriate choice of architecture pattern based on the unique needs of the project. To assist with these choices, we have explained the various cases and the recommendation approach for that case respectively. As it is evident that the monolithic architecture satisfies the basic requirements of a backend server compared to the microservices architecture that is complicated to develop, however, the microservices services gives better response time and low rate of errors during heavy traffic on the backend server. The Monolithic architecture results in having low minimum latency but also has a very high maximum latency during heavy traffic. The microservices architecture has a higher minimum latency compared to monolithic architecture but can handle heavy traffic with ease. Considering the cost, the complicated architecture of the microservices architecture along with its communication, configuration and multiple devices/virtual machines makes it more expensive than the monolithic architecture. Due to the rapid growth of information and technologies, it has led to exponential growth of not only industries, companies and organizations but also a massive number of users. To handle this growth, robust and high-performance systems needed so the development of various distributed systems such as the microservices architecture in web application development are in demand. Despite all the benefits and drawbacks it has brought with it, there is still a tremendous amount of room for improvement. The microservices architecture needs technical innovations to

give solutions to problems such as network complexity, security issues, etc.

## References

- [1] Raj, V. and Sadam, R. (2021) 'Performance and complexity comparison of service oriented architecture and microservices architecture', *Int. J. Communication Networks and Distributed Systems*, Vol. 27, No. 1, pp.100–117
- [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] O. Al-Debagy and P. Martinek, "A Comparative Review of Microservices and Monolithic Architectures," 2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI), 2018, pp. 000149-000154, doi: 10.1109/CINTI.2018.8928192.
- [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), 2020, pp. 150-153, doi: 10.1109/MEMSTECH49584.2020.9109514.
- [5] 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), 2020, pp. 1-4, doi: 10.1109/AICT50176.2020.9368571.
- [6] F. Ponce, G. Márquez and H. Astudillo, "Migrating from monolithic architecture to microservices: A Rapid Review," 2019 38th International Conference of the Chilean Computer Science Society (SCCC), 2019, pp. 1-7, doi: 10.1109/SCCC49216.2019.8966423.
- [7] G. Liu, B. Huang, Z. Liang, M. Qin, H. Zhou and Z. Li, "Microservices: architecture, container, and challenges," 2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C), 2020, pp. 629-635, doi: 10.1109/QRS-C51114.2020.00107.
- [8] Bogner J, Fritzsche J, Wagner S, Zimmermann A (2019) Assuring the evolvability of microservices: insights into industry practices and challenges. In: *Proceedings of the 2019 IEEE international conference on software maintenance and evolution (ICSME)*, pp 546–556
- [9] Fritzsche J, Bogner J, Wagner S, Zimmermann A (2019) Microservices migration in industry: intentions, strategies, and challenges. In: *Proceedings of the 2019*

IEEE international conference on software maintenance and evolution (ICSME), pp 481–490

- [10] Balalaie, A., Heydarnoori, A. and Jamshidi, P. (2016) ‘Microservices architecture enables devops: migration to a cloud-native architecture’, IEEE Software, 18 March, Vol. 33, No. 3, pp.42–52.
- [11] R. Chen, S. Li, and Z. Li, “From Monolith to Microservices: A Dataflow-Driven Approach,” in 2017 24th Asia-Pacific Software Engineering Conference (APSEC), 2017, pp. 466–475.
- [12] L. Carvalho, A. Garcia, W. K. G. Assunç ao, R. de Mello, and M. J. de Lima, “Analysis of the criteria adopted in industry to extract microservices,” in Proc. Joint 7th Int. Workshop Conducting Empirical Stud. Ind., 2019, pp. 22-29.
- [13] J. Jaworski, W. Karwowski, and M. Rusek, “Microservice-based cloud application ported to unikernels: Performance comparison of different technologies,” in Proc. 40th Anniversary Int. Conf. Inf. Syst. Archit. Technol., L. Borzowski, J. Więtek, and Z. Wilimowska, Eds. Cham, Switzerland: Springer, 2019, pp. 255-264.
- [14] M. Jagie<sup>aa</sup>o, M. Rusek, and W. Karwowski, “Performance and resilience to failures of an cloud-based application: Monolithic and microservices based architectures compared,” in Computer Information Systems and Industrial Management, K. Saeed, R. Chaki, and V. Janev, Eds. Cham, Switzerland: Springer, 2019, pp. 445-456.
- [15] G. Granchelli, M. Cardarelli, P. Di Francesco, I. Malavolta, L. Iovino, and A. Di Salle, “Towards recovering the software architecture of microservice-based systems,” in Proc. IEEE Int. Conf. Softw. Archit. Workshops (ICSAW), Apr. 2017, pp. 4653.