A Study on the Most Prominent Areas of Research in Microservices

Mohammad Sadegh Hamzehloui, Shamsul Sahibuddin, and Ardavan Ashabi

***Abstract*—Microservices have recently gained a lot of attention in the software industry. Their modularity and smaller size offer flexibility advantageous to both development and operational teams. However, the bigger picture is still lacking despite numerous researches on microservices. There are few aspects of microservices that have never been discussed in depth despite being acknowledged repeatedly. The current research is the continuation of our previous paper, "A systematic mapping on microservices". In the named paper we have identified the focus areas of microservices' researches. Along with our previous findings we have spotted several crucial key points that require further discussions. These includes: definition of microservices, their sizes and boundaries. We have also explored the relationship of microservices with SOA and DDD. These are the two terms that are frequently associated with microservices. Finally, we have discussed DevOps, cloud and virtualization as three of the most essential factors in microservices ecosystem. We attempted to clarify the role of each of these factors. Based on our findings, there is still no standardized definition for microservices to-date. In absence of clear guidelines, SOA and DDD concepts are widely being used to develop microservices. DevOps practices together with the cloud environment are playing an important role in facilitating the implementation of microservices. We have also identified containerization as an effective method to overcome the hardware limitation besides speeding up the delivery process.**

***Index Terms*—Microservices, definition, SOA, DDD, DevOps.**

I. Introduction

Previously, applications were developed using monolithic approaches. This means a single code-base was used to run an entire application. Designing a monolithic application has been known to be less complicated compared to the more distributed approaches. Running them on the other hand is not without challenges. Monolithic applications lack some cardinal features, namely scalability and flexibility [1]. The fact that monolithic applications are developed as one piece makes it difficult to alter. The difficulty arises from the high coupling nature of these applications. Scaling monolithic applications is another major obstacle due to their larger size. A monolithic application needs to be scaled in its entirety when only a certain segment of it needs to be scaled. These issues in addition to the rise of the new technologies has led the software industry to search for alternatives. Microservices are one of the solutions that addresses many of the aforementioned issues. Microservices are a set of services designed to work together to form an application. Each service is built to execute a single task [2]. Their smaller sizes ease the process of deployment and replacement. This enable companies to accomplish tasks that would have been otherwise difficult if not impossible, with the monolithic applications. A single service with a higher incoming traffic can be scaled while the rest of services are left at the same size. This will tremendously help the industries to reduce the cost of running services. Microservices are also compatible with the new DevOps (Development and Operations) culture. DevOps is emphasizing on shortening time from development to deployment by following practices such as automation, CD (Continuous delivery) and CI (Continuous Integration). Since microservices are relatively new in the software industry, there are many researches being conducted to improve its current understanding. There are wide range of papers that are focused on different aspect of microservices. Categorizing and analyzing these researches' findings will assist in grasping the bigger picture. This will also help to improve understanding the current state of the development of microservices. The current research is the continuation of our previous paper that was a systematic mapping on microservices [3]. In the previous paper, we attempted to identify the main areas of focus in researches conducted on microservices. In this paper we have summarized additional findings which we could not include in the previous paper due to word limitations.

We have organized this paper as the following: Section II is an overview of our previous paper. Section III lists down our research questions. Section IV explains the findings of the related papers. Section V is the main body of the paper. It discusses the areas of debate. Section VI is the conclusion.

II. Background

Our previous research [3] was a systematic mapping on microservices. We have selected 38 papers from three resources: IEEE, ACM and Scopus. The selected papers were subsequently categorized into three:

- Infrastructure
- Deployment and Management
- Software

In previous paper we have concluded that majority of the selected papers were in the infrastructure category. Within the infrastructure category, DevOps practices were among the highest covered topics. Architecture and cloud were the two most frequently used terms.

Manuscript received October 12, 2018; revised March 15, 2019.

Mohammad Sadegh Hamzehloui, Shamsul Sahibuddin, Ardavan Ashabi are with University Technology Malaysia, Malaysia (e-mail: ardavan.ashabi@gmail.com).

We went through numerous researches to write our previous paper. Many of those papers attempted to provide an overview of the current trends and focus areas in microservices. From our primary research it was clear that a set of tools and concepts are closely related to microservice. But many of these tools and concepts were very superficially discussed. Given the fact that microservices are a new concept in software engineering, we felt the need to further elaborate some of these concepts. More importantly, we believe that explaining the relationship between these concepts and how they each fit into microservices' ecosystem will provide further clarity. This current paper is the extension of our previous paper, due to restriction in the word count and the specificity of the topic.

III. Research Questions

Q1. What are the most prevalent areas of debate surrounding microservices?

Answering this question will cover most of the undergoing research. By narrowing down the important areas of debate, we are hoping to establish a foundation for upcoming researches. We are attempting to concentrate more on concepts we found ambiguous.

Q2. What are the common technologies/practices used in developing and running microservices?

Answering this question will provide a generic overview for anyone who is conducting a research on microservices. This will also be helpful for industries that are planning to use microservices in assisting them to be familiar with the current trends of microservices. Although there have been researches addressing similar questions, it is necessary to conduct these researches on a regular basis in order to keep up with the latest updates given that microservices are still evolving.

IV. Related Works

In our previous paper we have briefly covered 5 similar researches. Three of which were in form of systematic mapping. One of them was a literature review and the last one a taxonomy of microservices. Since we have changed the scope of the research, additional 2 papers will be discussed as related works. It is important to mention that most of researches we found were published after the year of 2014. This indicates that microservices started to become a trend after 2014. This is also similarly reflected by the findings of the other two systematic mappings [4], [5].

The first systematic mapping is this paper [6] which conducted its research on 21 selected papers was published between 2014 and 2015. Their research indicated that the most common type of research was solution proposal. The validation researches were at the second place, followed by experiment and evaluations papers. Their results are quite similar to our findings with the exception of evaluation researches. Based on our findings, the number of evaluation and validation researches were equal. Increasing the number of evaluation papers can be an indication that microservices and their related technologies are now more commonly implemented in practice. Fig. 1 demonstrates the results of our findings on the research types of selected papers. They also concluded that microservices' migration, autonomous healing, DevOps and autoscaling are some of the common trends.

The next systematic mapping was on microservices' architecture[4]. They selected 33 articles to conduct their research including papers from the year of 2016. Based on their findings, evaluation researches followed by solution proposals were the most frequent research types. Their findings suggest that modularity, scalability, and maintainability are among the most discussed attributes. They also highlight security as one of the least discussed attributes. Concurring with the previous paper, they believe that DevOps practices are the future trends.

The next related paper was a systematic mapping with a focus on trends in microservices' architecture [5]. They included 71 papers including many from 2016. Based on their findings, solution proposals were the most common type of researches. Validation and opinion papers were second and third. Evaluation researches were the least common. They suggested that this implies lack of papers written on real-world industrial experiments. They argued that complexity is the most recurrent problem that have been discussed. Low flexibility followed by management and service composition are the next frequently discussed topics. They also pointed out that security is still a topic that is not covered by many researches. Listed below are some of their other findings:

- A close relation between microservices architecture, DevOps and cloud
- Containerization and virtualization are the key enabling technologies

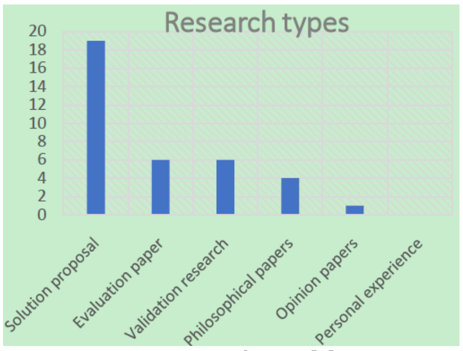

Fig. 1. Research types [3].

- Performance efficiency, maintainability and security are the at the top of quality metrics
- Monitoring, system-level management and service orchestration are the top three main infrastructural related discussions.

Another paper that provides a valuable insight into microservices is a taxonomy of microservices conducted by Martin Garriga [7]. They did a thorough study on different aspects of microservices by providing a great overview of different aspects of microservices. Their research was significant in explaining the relationship between different aspects of microservices. They came out with a sequence of tables that demonstrates the main categories and subcategories of the microservices' related topics. They covered 28 papers published up to 2016. Below is a list of some of the key points from their paper:

- Security is a topic that is not covered extensively.

- Design practices and patterns are not adequately discussed.
- Finding the right granularity level for microservices is still an issue.
- RESTful HTTP is the most common method of communication

The next paper is a systematic literature review conducted on microservices [8]. They conducted their research on 37 papers. There were a few interesting key points in their research. They attempted to list down the characteristics of microservices. Lack of a standardized definition for microservices is a known problem in the software community, which we will discuss in the next section. Hence their attempt can be viewed as a good starting point in forming a standard definition for microservices. They listed some of the emerging technologies. REST topped the list followed by swagger. They have also come out with the list of the most common technologies that are currently being used. Docker is at the top of this list with Node.js and MySQL being the second and third.

Microservice Design for Container based Multi-cloud Deployment [9] is another paper that attempted to explain different aspects of microservices. In their attempt to bring standardization to microservices, they managed to provide a comprehensive explanation on microservices' architecture. They have discussed some of the pros and cons of using microservices and explained some of the microservices' design patterns. What grabbed our attention was their inclusive list and explanation of the microservices characteristics provided. They listed down seven characteristics of microservices. We will be discussing some of them in the next section.

The last paper that we found relevant to our topic is this paper [10]. The significance of this papers is the comparison that they made between microservices' architecture and SOA. SOA and microservices have been discussed in numerous papers. The distinction between the two however, have rarely been discussed explicitly. Another significance of this paper is their attempt to define microservices. Although such attempts might have been made in other papers, it is the depth of the discussion, that distinguishes the paper.

V. Areas of Debate

Since microservices are a relatively new technology, there are still many debates surrounding them. These ranges from design perspectives to quality and performance metrics to implementation and maintenance. After going through many papers we found that certain areas are more frequently debated than others. This may be caused by both their importance and existing ambiguity surrounding them.

A. Definition of Microservices' Architecture

Microservices' architecture have been the topic of many researches, and yet they lack a precise definition [11], [12]. "Building microservices" was one of the first books written on the topic of microservices. Sam Newman in his book attempted to define microservices. He described them as:

"Microservices are small, autonomous services that work together."[13]

Another attempt was made by the authors of microservice architecture book [14]:

"A microservice is an independently deployable component of bounded scope that sup- ports interoperability through message-based communication. Microservice architecture is a style of engineering highly automated, evolvable software systems made up of capability-aligned microservices."

To overcome the issue of defining microservices, some researchers attempted to identify and categorize their common architectural characteristics.

Although there are occasional differences in their listed characteristics, majority of papers have identified similar attributes. Some of the most used terms to describe the microservices architecture are:

- Small in size [4] , [12], [13]
- Single-responsibility [9], [15]
- Loosely coupled [6], [9], [16], [17],
- Explicitly published interfaces [9], [18]
- Lightweight [6]
- Communicating via REST and HTTP [9], [19]

Communicating via REST and HTTP are listed by many papers as one of the microservices' characteristics, however that is not always the case. REST APIs are indeed one of the most widely used technologies in microservices, but there are other alternatives to REST, such as RPCs.

B. Identifying the Size and Boundaries

Defining the right size and boundaries for microservices is another debatable area. Few metrics have been used to measure the size of microservices. Some of them are [20]:

- The number lines of the codes used to develop microservices
- Being able to rewrite a microservice within 2-6 weeks period
- Having a two-pizza team (Enough to feed the whole team) [13]

These measures are clearly crude methods for defining the size of microservices. For instance, different programming languages may use more or less lines of codes to execute the same task, or the complexity of a service may change its refactoring time. Considering the fact that the size of microservices can greatly affect some aspects like performance, it is crucial to define new measures [20].

It is also important to define the boundaries of microservices. This means that there should be a clear division of which services implementing what functionalities. Defining these boundaries requires a set of tools and experience that we currently lack [21], [22]. This is probably the reason that DDD and SOA are gaining popularity. DDD due to its ability to define bounded contexts that can be correlated to microservices [23], and SOA due to its modularity and implementation similarities with microservices.

C. SOA and DDD

The term microservices have been repeatedly associated with two other terms, SOA and DDD. SOA or Service Oriented Architecture is a design approach where multiple services together provide a set of capabilities. SOA was initially developed as a solution to overcome the issues with monolithic applications [13]. Reusability [24] and loose coupling are the fundamental principles of SOA [25]. SOA is made of several components. The main ones are:

- SOA services
- EBS or Enterprise Service Bus
- Service management and monitoring

Sam Newman in his book [13] describes the problems of SOAs as:

- communication protocols
- vendor middleware
- lack of guidance about service granularity

He also believes that microservices are the result of our better understanding of SOA, so we should think of them as a specific approach for SOA. Some other papers on the other hand describe microservices as a broader area of SOA with focus on some of its specific aspects, such as agile and DevOps practices [5]. Despite the similarities between microservices and SOA there are differences between the two. Microservices eliminate the need for ESB (Enterprise Service Bus) by using "smart end-points" [24]. Independence is the key characteristic of microservices [25]. A microservice should be independently deployable. Microservices architecture promote the idea of share-nothing philosophy, whereas SOA promotes the idea of share-as-much-as-you-can [5]. SOAs are designed to be loosely-coupled as oppose to microservices that are un-coupled. While SOAs are designed to be cohesive, microservices are designed to independent. Finally, SOAs are following a centralized governance design. On the other hand, microservices' governance is decentralized [25].

DDD a software development approach with a focus on application domains and their relationship has gained more popularity with the emergence of microservices [26]. This approach has been adopted by many who are trying to implement microservices. This is mostly due to its well-established set of practices that enables modeling complex systems [14]. DDD introduces the concept of bounded context. Each component in the system only exists within its bounded context. They can also represent business domains. Following the concept of bounded contexts [27] is a good way to start with designing microservices. This ensure that the designed microservices are loosely coupled [13], [14].

D. DevOps

DevOps is another term that is used often with microservices. DevOps is a set of techniques that aims to unify software development techniques with deployment and operations [28]. DevOps and microservices are partially dependent on each-other. They are both relying heavily on cloud and virtualization [29].

There are many challenges that DevOps is trying to address, some of them are listed below [28]:

- Scaling
- Automation
- Continuous delivery
- Dealing with cloud technologies
- Monitoring
- Resource efficiency

But how microservices and DevOps are related? Microservices are the solution to some of these problems, namely continuous delivery and scaling. Due to their smaller size, they shorten the delivery process and make scaling less challenging [21]. On the other hand, to use microservices, a highly automated system with an advanced monitoring capability is required [19], [30]. A such system is advocated by DevOps. Cloud is an environment that ease the processes of implementing a system with these characteristics.

E. Cloud

Cloud provides many of the functionalities required to implement microservices. It is also an ideal environment to apply DevOps practices. It is built based on a set of loosely coupled components. This makes microservices' structure fit perfectly to the elasticity of the cloud. Cloud services provide dynamic and scalable solutions with more efficient resource usage. These features make the cloud a more prominent environment to run microservices [31]. Having a wide range of configurations enables cloud users to configure their infrastructure exactly according to their applications' requirements [28]. Cloud services are readily accessible via web, and that makes them universally available. This enables its users to manage them from any geographical location [32].

Many cloud services are providing features such as IaaS (Infrastructure as a Service), PaaS (Platform as a Service), SaaS (Software as a Service). These features enable developers to customize their infrastructure and platform mostly by coding. And lastly, cloud platforms allow to pay on-demand. This will provide a great flexibility to add the required hardware as the application scales up or down. This can help the users to minimize the running cost [33]. Cloud providers are also offering a set of tools to enhance monitoring, and automation processes. This will ease the process of implementing DevOps practices. Cloud services are using virtualization to provide isolated environments and overcome hardware limitations.

F. Virtualization

Virtualization platforms allow us to provision and resize our machines at will, with infrastructure automation giving us a way to handle these machines at scale [13].

Virtualization is a common solution to tackle hardware limitations. It is also the way that cloud services provide scalable solutions to their clients. Besides scalability, isolation is another reason for using virtualization. A lot of the software companies are using some type of virtualization to separate different development environments. This separation makes it easier to do the development, testing and SQA (software quality assurance). Virtualization is also helpful to implement continuous development and DevOps (Development and Operations), which are the critical parts of deploying microservices. There are two main virtualization types commonly used by industries when it comes to microservices; VMwares and containers. VMwares are the traditional method of virtualization that have been around for years. Containers on the other hand, are a new trend despite being around for a long time. Their recent popularity is due to the new software named Docker.

Docker is a new software that is taking advantage of the Linux containerization feature to create an isolated environment. Using a Unix built-in feature makes Docker superior to traditional VMware when it comes to performance. As Stubbs, Walter, Dooley [34], explained it:

"While virtual machines provide an abstraction at the hardware level, the container model virtualizes operating system calls so that typically many containers share the same kernel instance. This key difference makes for a much more lightweight virtualization package that can be built, modified, rebuilt and shared far more rapidly than its hypervisor-based predecessor. "

There are different architectural designs for implementing microservices. In some of these designs, different services are required to work on the same machine but acting as an independent service. Virtualization tools such like Docker are amongst the best solutions to implement these architectures. This is attributed to their light weight and the level of isolation that they provide [1].

G. Docker

Docker is currently the most popular container solution [14], [35]. There are a lot of different technologies built around Docker [36]. This makes Docker a viable option for companies that decided to use containers to run microservices. They are known to facilitate processes like rolling updates and automated scaling [37].

Docker comes with its own ecosystem [36]. In fact, many of the cloud service provides are offering Docker as a part of their implementational system. Listed below are some of the main characteristics of Docker that makes it a popular tool for running microservices:

1) Light weight

The small size of containers helps to reduce resource consumption. The smaller size also accelerates the automated processes and continuous delivery [1].

2) Dockerfile

Dockerfiles are the list of required commands to build Docker images. They can be stored in the form of a single text file. This makes it possible to version control them.

3) Docker images

Docker images are built based on the instructions in the Dockerfiles. They can be shared within the community using Docker repositories. These repositories can be both public and private.

4) Docker ecosystem

There are many applications built around Dockers. They range from monitoring tools to management and repositories.

VI. Conclusion

In this paper we attempted to identify and discuss the main areas of debate in microservices. We have also tried to identify the main technologies and practices involved in running microservices. The identified areas were based on our previous experience and the detailed systematic mapping that we have recently conducted. The definition of microservices and identifying the boundaries of them are the two areas requiring further research. In the absence of a standardized definition, we listed down few of the main characteristics of microservices based on the literatures available. We have also concluded that new metrics are required to design microservices. These metrics should address the issue of the size and boundaries of microservices. The absence of such metrics is probably the reason that many are using the other existing approaches to fill this gap. SOA and DDD are currently the closest software styles to microservices. Many are using their concepts to develop microservices. Although they are a good place to start with, specialized solutions should be designed with the focus on microservices.

We have also discussed DevOps, cloud and virtualization and their relationship with microservices. We believe that these three components are essentials for implementing microservices and vice versa with the exception of virtualization. Although microservices might not be dependent on them, but they will unleash the true power of microservices. Practices like DevOps might be very difficult to follow using traditional application development methods. Using clouds' distributed components will be much easier while taking advantage of microservices.

References

[1] D. Jaramillo, D. V. Nguyen, and R. Smart, "Leveraging microservices architecture by using Docker technology," *SoutheastCon*, 2016.

[2] A. D. Camargo, I. Salvadori, R. Mello, S. Dos, and F. Siqueira, "An architecture to automate performance tests on microservices," in *Proc. 18th Int. Conf. Inf. Integr. Web-based Appl. Serv*, 2016.

[3] M. S. Hamzehloui, S. Sahibuddin, and K. Salah, "A systematic mapping study on technical debt," in *Proc. 3rd International Conference of Reliable Information and Communication Technology*, pp. 1–12, 2018.

[4] N. Alshuqayran, N. Ali, and R. Evans, "A systematic mapping study in microservice architecture," in *Proc. 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, 2016, pp. 44–51.

[5] D. P. Francesco, I. Malavolta, and P. Lago, "Research on architecting microservices: Trends, focus, and potential for industrial adoption," in *Proc. 2017 IEEE Int. Conf. Softw. Archit*, 2017.

[6] C. Pahl and P. Jamshidi, "Microservices: A systematic mapping study," in *Proc. 6th Int. Conf. Cloud Comput. Serv. Sci*., 2016, pp. 137–146.

[7] M. Garriga, "Towards a taxonomy of microservices architectures," Springer, 2018.

[8] H. Vural, M. Koyuncu, and S. Guney, "A systematic literature review on microservices," *Int. Conf. Comput. Sci. Its Appl.*, 2017.

[9] K. Y. B. Jambunathan, "Microservice design for container based multi-cloud deployment microservice design for container based multi-cloud deployment," *Jour Adv Res. Dyn. Control Syst*, 2017.

[10] D. Shadija, M. Rezai, and R. Hill, "Towards an understanding of microservices," in *Proc. 2017 23rd IEEE Int. Conf. Autom. Comput. Addressing Glob. Challenges through Autom. Comput*, 2017, pp. 7–8.

[11] M. F. J. Lewis. Microservices, a definition of this new architectural term. [Online]. Available: https://martinfowler.com/articles/microservices.html

[12] T. Asik and Y. E. Selcuk, "Policy enforcement upon software based on microservice architecture," in *Proc. 2017 15th IEEE/ACIS Int. Conf. Softw. Eng. Res. Manag. Appl.*, 2017.

[13] S. Newman, *Building Microservices*, Sam Newman, 2015.

[14] I. Nadareishvili and R. M. M Mitra, *Microservice Architecture*, 2016.

[15] M. Ahmadvand and A. Ibrahim, "Requirements reconciliation for scalable and secure microservice (de)composition," in *Proc. 2016 IEEE 24th Int. Requir. Eng. Conf. Work*., 2016.

[16] A. R. Sampaio, H. Kadiyala, B. Hu, J. Steinbacher, T. Erwin, N. Rosa, I. Beschastnikh, and J. Rubin, "Supporting microservice evolution," in *Proc. 2017 IEEE Int. Conf. Softw. Maint. Evol*, 2017.

[17] P. Wizenty, J. Sorgalla, F. Rademacher, S. Sachweh, *Archit. Companion*, 2017.

[18] S. Haselbock and R. Weinreich, "Decision guidance models for microservice monitoring," in *Proc. 2017 IEEE Int. Conf. Softw. Archit. Work.*, pp. 54–61, 2017.

[19] N. Ashikhmin, G. Radchenko, and A. Tchernykh, "RAML-based mock service generator for microservice applications testing," *Commun. Comput. Inf. Sci.*, vol. 793, pp. 456–467, 2017.

[20] S. Klock, V. D. Werf, J. M. E. M. Guelen, and J. P. S. Jansen, "Workload-based clustering of coherent feature sets in microservice architectures," in *Proc. 2017 IEEE Int. Conf. Softw. Archit*, 2017.

[21] L. Chen, "Microservices: architecting for continuous delivery and devops," in *Proc. 2018 IEEE Int. Conf. Softw. Archit.*, 2018, pp. 39–397.

[22] S. Hassan, N. Ali, and R. Bahsoon, "Microservice ambients: An architectural meta-modelling approach for microservice granularity," in *Proc. 2017 IEEE Int. Conf. Softw. Archit.*, 2017. , pp. 1–10

[23] E. Schäffer, H. Leibinger, A. Stamm, M. Brossog, and J. Franke, "Configuration based process and knowledge management by structuring the software landscape of global operating industrial enterprises with Microservices," *Procedia Manuf.*, vol. 24, pp. 86–93, 2018.

[24] Y. Yu, H. Silveira, M. Sundaram, Y. Yale, H. Silveira, and M. Sundaram, "A microservice based reference architecture model in the context of enterprise architecture," in *Proc. 2016 IEEE Adv. Inf. Manag. Commun. Electron. Autom. Control Conf.*, 2016, pp. 1856–1860.

[25] Z. Xiao, I. Wijegunaratne, and X. Qiang, "Reflections on SOA and microservices," in *Proc. 4th Int. Conf. Enterp. Syst. Adv. Enterp. Syst*, 2016, pp. 60–67.

[26] F. Rademacher, S. Sachweh, and A. Zündorf, "Towards a UML profile for domain-driven design of microservice architectures," LNCS, pp. 230–245, 2018.

[27] J. Bonér, *Reactive Microservices Architecture*, 2016.

[28] H. Kang, M. Le, and S. Tao, "Container and microservice driven design for cloud infrastructure devops," in *Proc. 2016 IEEE Int. Conf. Cloud Eng.*, 2016, pp. 202–211.

[29] C. Barna, H. Khazaei, M. Fokaefs, and M. Litoiu, "Delivering elastic containerized cloud applications to enable devops," in *Proc. 2017 IEEE/ACM 12th Int. Symp. Softw. Eng. Adapt. Self-Managing Syst.*, 2017, pp. 65–75.

[30] D. Taibi, V. Lenarduzzi, and C. Pahl, "Architectural patterns for microservices: A systematic mapping study," in *Proc. 8th Int. Conf. Cloud Comput. Serv. Sci.*, 2018, pp. 221–232.

[31] E.Christian and A. Castiglione, *Challenges in Delivering Software in the Cloud as Microservices*, 2016.

[32] S. Suram, N. A. MacCarty, and K. M. Bryden," Engineering design analysis utilizing a cloud platform," *Adv. Eng. Softw.*, vol. 115, pp. 374–385, 2018.

[33] M. Miglierina, "Application deployment and management in the cloud," in *Proc. 2014 16th Int. Symp. Symb. Numer. Algorithms Sci. Comput.*, 2014, vol. 7, pp. 422–428.

[34] J. Stubbs, W. Moreira, and R. Dooley, "Distributed systems of microservices using docker and serfnode," in *Proc. 7th Int. Work. Sci. Gateways*, 2015.

[35] T. Hunter, *Advanced Microservices*, 2017.

[36] D. Liu, H. Zhu, C. Xu, I. Bayley, D. Lightfoot, M. Green, and P. Marshall, "An integrated development environment for microservices," in *Proc. 2016 IEEE Int. Conf. Serv. Comput. CIDE*, 2016.

[37] R. Heinrich, A. V. Hoorn, H. Knoche, F. Li, L. E. Lwakatare, C. Pahl, S. Schulte, and J. Wettinger, "Performance engineering for microservices: Research challenges and directions," in *Proc. 8th ACM/SPEC Int. Conf. Perform. Eng. Companion*, 2017, pp. 223–226.

Mohammad Sadegh Hamzehloui holds a bachelor of computer security and a MSc in software engineering from University of Staffordshire. He is currently pursuing his Ph.D at University of Technology Malaysia. His research interests are software architecture and design, distributed systems and cloud native applications.

Shamsul Sahibuddin is currently a professor at Universiti Teknologi Malaysia (UTM). Prof. Dr. Shamsul is the dean of Advanced Informatics School at Universiti Teknologi Malaysia, Malaysia. He graduated from Aston University, United Kingdom with a Ph.D. in computer science.

Ardavan Ashabi has been a PhD student in computer science at University Technology Malaysia (UTM) since 2014 focusing on big data analysis algorithms. He obtained his MSc in computer science from Advanced Informatics School (AIS) at UTM in 2013 focusing on, system architecture and algorithm design. He got his degree in 2008 in software engineering. He has conducted research on big data, algorithm design, system architecture and cloud computing.