2016 IEEE 9th International Conference on Service-Oriented Computing and Applications

A Systematic Mapping Study in Microservice Architecture

Nuha Alshuqayran, Nour Ali and Roger Evans
Computing Engineering and Mathematics
University of Brighton
Brighton, UK
{n.alshuqayran, n.ali2, r.p.evans}@brighton.ac.uk

***Abstract*—The accelerating progress of network speed, reliability and security creates an increasing demand to move software and services from being stored and processed locally on users' machines to being managed by third parties that are accessible through the network. This has created the need to develop new software development methods and software architectural styles that meet these new demands. One such example in software architectural design is the recent emergence of the microservices architecture to address the maintenance and scalability demands of online service providers. As microservice architecture is a new research area, the need for a systematic mapping study is crucial in order to summarise the progress so far and identify the gaps and requirements for future studies. In this paper we present a systematic mapping study of microservices architectures and their implementation. Our study focuses on identifying architectural challenges, the architectural diagrams/views and quality attributes related to microsevice systems.**

I. Introduction

A. The microservices architecture

The microservices architecture has become a dominant architectural style choice in the service oriented software industry. Microservices is a style of architecture which puts the emphasis on dividing the system into small and lightweight services that are purposely built to perform a very cohesive business function, and is an evolution of the traditional service oriented architecture style [23]. It is also defined in [8] as "the minimal independent process that interact via messaging", and microservice architecture as "a distributed application where all its modules are microservices". The commonly agreed on benefits of this style include: increase in agility, developer productivity, resilience, scalability, reliability, maintainability, separation of concerns, and ease of deployment. However, those benefits come with challenges, such as discovering services over the network, security management, communication optimization, data sharing and performance. When addressed appropriately, however, these challenges allow a system to benefit from most of the aforementioned benefits [40].

definition

challenges

Over the last decade, leading software consultancy firms and product design companies have found the microservices approach to be an appealing architecture that allows teams and software organizations to be more productive in general, and build frequently more successful software products. Many organizations outside of the traditional software business have also tried and tested this style of architecture and have found it to be very beneficial. Microservices is also considered as an appropriate architecture for systems deployed on cloud infrastructures, as it can take advantage of the elasticity and on-demand provisioning features of the cloud model. Companies such as Netflix, and SoundCloud have adopted the microservices style in the cloud and gained many benefits from it [42] [6].

B. The need for a systematic mapping study

Even though microservices have emerged from the software industry and have been the focus of practitioners in the last decade[28][22], academic researchers have not kept with the pace. Only recently, they have started investigating this approach and providing original research to support it, such as new methodologies, processes and tools [28]. The motivation of this mapping study has its basis in the lack of available studies regarding the research performed for the microservices style. We have encountered one such study published in [31], however the study was limited in providing a temporal overview of microservice research.

As microservices is an architectural style, the objective of our study is to explore how previous research has supported microservices through architectural approaches. Secondly, our study will follow a characterization framework that is based on microsevice architectural challenges. The aim of the study is to focus on the proposed research questions, closely link and correlate the research questions to the mapping study results and provide quantified evidence from the available publications. Also, we will attempt to discover any specific areas of the microservices-style architecture that have not been explored yet and identify the areas where there is lack of published research.

The rest of this paper is structured as follows: Section II explains the research method that we have followed in this study. Section III illustrates the results, in terms of stating the popularity of microservice challenges in the selected studies, architecture views/diagrams and quality attributes of the microservices architecture. A discussion of these results follows in section IV, then we finally conclude and summarize all the outcomes of this exercise in section V.

978-1-5090-4781-9/16 $31.00 © 2016 IEEE
DOI 10.1109/SOCA.2016.15

II. Research Method

Systematic mapping studies are considered comprehensive and rigorous reviews of specific research questions in an area or a topic, which aim to identify the gaps in the literature and identify where new or better primary studies are needed to be put in place [18]. In this paper, a systematic mapping study in "Microservices-style Architecture" is presented by following the guidelines outlined in [5], [33], [18] and [9].

A. Planning stage

Initially, a set of research questions were drafted for investigation during the study. The motivation behind each research question was reviewed and refined. Subsequently, selected papers were assessed against quality criteria and a classification scheme was iteratively developed closely following a synthesis method. In summary, the review was established by conducting the following steps consecutively:

1) Research questions: The research questions and the motivations are outlined in Table I.

2) Search Strategy: The terms "microservice", "micro service", "micro-service" and "microservice architecture" were researched in articles published in journals, conferences and workshops. Sources from books, thesis, talks and blog posts were excluded. The research was restricted to articles published between 2014 and 2016, as there was no consensus on the term microservice architecture in the field prior to that date, according to [31]. Three online libraries were used IEEExplore, ACM DL and Scopus (which includes Springer).

3) Selection of primary studies: Before selection, articles were initially cross-checked for relevance against the research questions that are related to this systematic mapping study. The titles, abstracts and keywords were scanned to determine the relevance of the articles and whether they should be included or excluded for the purpose of this study, based on the criteria listed in Table II. After applying the exclusion and inclusion criteria, a total of 33 articles were collected in this study. Table III lists all the selected publications.

4) Keywording and Classification: Once papers were selected, a qualitative assessment was conducted to create an outline model for the quality of work. This helps to abstract various possible dimensions for characterization and categorization. As a result, the research classification approach performed in [46] was found to be generally applicable for this research and was used to classify the papers as: Evaluation Research, Opinion Paper, Solution Proposal, Experience Paper, Validation Research and Philosophical Paper.

A) Identified keywords for microservices challenges (RQ1):

Subsequent to the first round of review, the following keywords were identified to be mapped and linked to the challenges of creating microservices style systems. Keywords associated with challenges were identified to answer the study research question 1.

- Communication/Integration: Communication and integration have many facets to them in a microservices-style architecture. Defining a correct communication strategy is vital to the design. The strategy involves identifying the right protocol, response time expectations, timeouts and API design.
  Keywords: API, REST, sockets, TCP, gateway, circuit breakers, load balancer, proxy
- Service discovery: This is the ability of various services to discover each other in a consistent manner. It is important for a system to have a standard and consistent process for which services can register and announce themselves. This help the consuming services to discover the end points and the locations of other services. It also involves deciding the right consumer strategy and specifying how API gateways are configured to report service availability and discovery.
  Keywords: discovery, registration, service registry
- Performance: It was commonly observed that introducing microservices architecture to the software industry often adds more 'chatty' communication between the different services. For example, fulfilling one single business functional requirement would result in orchestrating multiple service calls together, which in return introduces additional lag to the end-user experience. Due to bounded contexts, often data that is frequently used by a single microservice is owned by another. This requires creating data sharing and synchronization primitives to avoid the communication overhead caused by data copying which happens during the service invocations. The following keywords are normally associated with performance challenges.
  Keywords: QoS, performance, SLA, speed, simulation.
- Fault-tolerance: This is the ability of a system to recover from a partial failure. It is up to microservice developers to take that into consideration and provide proper mechanisms to gracefully recover or stop any failure to cascade or migrate to other parts of the system. This is normally expected in cloud environments where Infrastructure as a Service (IaaS) causes inevitable failures.
  Keywords: fault, failure, recovery, tolerance.
- Security: Security is a major challenge that must be carefully thought of in microservices architecture. Services communicate with each other in various ways creating a trust relationship. For some systems, it is vital that a user is identified in all the chains of a service communication happening between microservices. OAuth and OAuth2 are well-known solutions that are employed by designers to handle security challenges.
  Keywords: secure, authentication, authorization, OAuth, OAuth2, encryption, vulnerability, attack
- Tracing and Logging: In microservices-based systems, the services tracing and central logging are vital for developers to understand the system behavior as a whole. Breaking monolithic systems into microservices uses techniques that are traditionally employed for debugging and profiling systems. Various techniques and solutions are emerging to solve this problem. Distributed tracing is the ability of a system to track a chain of service calls

Table I
THE RESEARCH QUESTIONS AND THEIR MOTIVATIONS

Number	Research Question	Motivation
1	What are the architectural challenges that microservices systems face?	The aim is to explore all the published studies that were relevant to microservices to highlight the gaps in them and look for future solution foundations
2	What architectural diagrams/views are used to represent microservices architectures?	The aim is to identify and investigate what are the possible methods and models to best describe different aspects and levels of microservices architecture.
3	What quality attributes related to microservices are presented in the literature?	The aim is to recognize and disclose the gaps in current research and hence set the direction for future research.

Table II
THE SELECTION CRITERIA

	Criteria
Inclusion	• Studies presenting the definition of microservices architecture. • Studies that focus on microservices architecture and implementation. • Studies that focus on a platform to run systems following a microservices-style architecture. • Studies that focus on a specific challenge within microservices (e.g. fault tolerance, acceptance testing etc). • Studies that implement microservices-style architecture for a specific business or technical domain. • Studies that do comparisons between monolithic and microservices architectures.
Exclusion	• Papers using the microservice term but not to refer to the architectural style. • Papers which do not have real data to back the proposed design/methodology/architecture. • Studies that do not have microservices as their primary research topic or analysis. • Studies that focus on platforms that are not primarily designed to run microservices but may allow it.

to identify a single transaction or a single user request. Logging is another critical component of any system. Logs are important for auditing and debugging purposes. Special attention must be paid to carefully design a central logging and aggregation system for developers to continue debugging systems in an appropriate manner.
Keywords: tracing, logging, debugging, profiling

- Application Performance Monitoring: APM is an infrastructural centric characteristic. It involves measuring individual microservices' performance to assess the health and existing SLAs for a system.
  Keywords: monitoring, APM, health monitoring
- Deployment operations: Deployment operations and scaling are fundamental infrastructure concerns. However, selecting the right platform influences significantly the architecture of a microservices system. Container orchestration tools and structured PaaS solutions provide various features that makes deployment and operations very trivial activities. However, selecting the right solution is critical as all of these platforms come with their own set of assumptions and opinions, which the designer has to follow to be able to utilize the selected platform to its potential. Scaling microservices can become a challenge if the right architecture is not followed. There are several guidelines such as 12 factor application and cloud native designs [1], which need to be followed to make service scaling easier. Most of these guidelines demand statelessness and portability by decoupling service runtime from OS and platform resources.
  Keywords: operations, orchestration deployment, scaling, auto-scale, rolling upgrades, images, container

5) Data extraction strategy and Quality Assessment : Data for this study was extracted using a machine learning-based PDF Extraction Library called grobid [10], which extracts the PDF data into structured TEI-encoded documents, with a particular focus on technical and scientific publications. [16] was used to do the visual analysis and generate various charts. First, the selected keywords for the 'challenges' part were analyzed using Kibana [16] visualization, in order to understand the distribution over the underlying population. This gave a quantitative indication of the possible classifications.

Furthermore, the studies have been subjected to a questionnaire to roughly classify the selected papers based on the actual content review. The research questions were the driving element of the questionnaire.

III. RESULTS

Significant Keywords:

At a high level, the following are the top significant keywords from the previously mentioned keywords lists. Figure 1 lists the top terms found in the literature. It can be observed that "deployment", "cloud" and "performance" are the words that dominate the papers, "deployment" is the most discussed topic appearing in 31 out of 33 papers, followed by "cloud" and "performance" in 23 papers.

Challenges of microservices system architecture (RQ1):

We identified papers which are actively addressing one or more of the challenges mentioned in "Identified Keywords for microservices challenges" (above). The classified papers either present a solution, addressing a challenge as the primary or secondary topics or discuss a challenge to a certain depth. Furthermore, we quantitatively searched for earlier presented keywords associated with the challenges in the papers and presented the count of papers mentioned in one or more of

[1] http://12factor.net/

Table III
PUBLICATIONS SELECTED

ID	Paper Name	Format
1	Sustaining runtime performance while incrementally modernizing transactional monolithic software towards microservices [19]	Conference
2	The hidden dividends of microservices [17]	Journal
3	An architecture for self-managing microservices [41]	Workshop
4	Synapse: a microservices architecture for heterogeneous-database web applications [44]	Conference
5	A reference architecture for real-time microservice API consumption [11]	Workshop
6	Exploring the impact of situational context: a case study of a software development process for a microservices architecture [30]	Workshop
7	Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud [45]	Conference
8	Microservice-based architecture for the NRDC [21]	Conference
9	Container and microservice driven design for cloud infrastructure DevOps [15]	Conference
10	Scalable microservice based architecture for enabling DMTF profiles [25]	Conference
11	Experience on a microservice-based reference architecture for measurement systems[43]	Conference
12	Microservice based tool support for business process modelling [1]	Workshop
13	Designing a smart city Internet of Things platform with microservice architecture [20]	Conference
14	Microservices [40]	Journal
15	A reusable automated acceptance testing architecture for microservices in behavior-driven development [34]	Conference
16	Microservices architecture based cloudware deployment platform for service computing [12]	Conference
17	Security-as-a-Service for microservices-based cloud applications[39]	Conference
18	CYCLOPS: A micro service based approach for dynamic rating, charging and billing for cloud [32]	Conference
19	Microservices validation: Mjolnirr platform case study [36]	Conference
20	Data-Driven workflows for microservices: genericity in Jolie[35]	Conference
21	Distributed systems of microservices Using Docker and Serfnode [38]	Workshop
22	Location and context-based microservices for mobile and Internet of Things workloads[4]	Conference
23	Performance evaluation of microservices architectures using containers [2]	Conference
24	CIDE: an integrated development environment for microservices[24]	Conference
25	Microservices and their design trade-offs: a self-adaptive roadmap [13]	Conference
26	SeCoS: Web of Things platform based on a microservices architecture and support of time-awareness[47]	Journal
27	Apache airavata as a laboratory:aArchitecture and case study for component-based gateway middleware [26]	Workshop
28	Microservices validation: methodology and implementation[37]	Workshop
29	Learning-based testing of distributed microservice architectures: Correctness and fault injection [27]	Conference
30	Automated deployment of a microservice-based monitoring infrastructure [7]	Journal
31	A microservice approach for near real-time collaborative 3D objects annotation on the web [29]	Conference
32	Multi cloud deployment with containers [14]	Journal
33	Migrating healthcare applications to the cloud through containerization and service brokering [3]	Conference

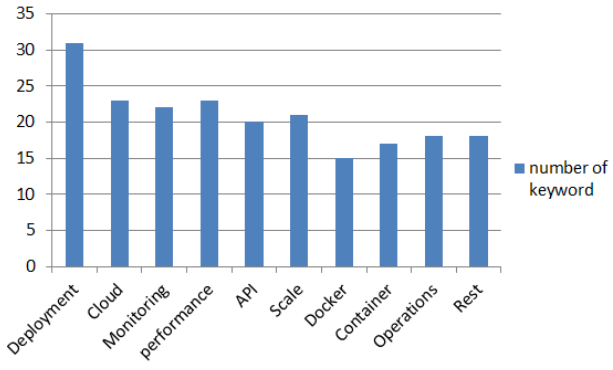

Figure 1. The top 10 keywords in the literature

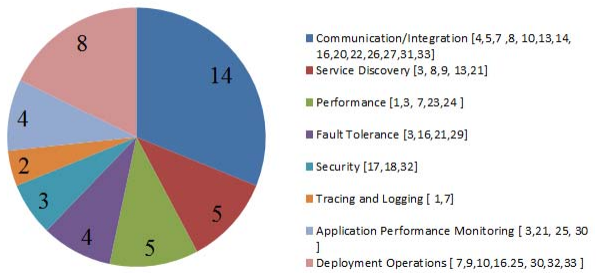

Figure 2. The distribution of microservices challenges in the literature

such keywords. Table IV and Figure 2 show the results of the above classification.

Research paper approaches:

We classified papers using approaches presented in Wieringa [46]. Since the microservices architectural style is an emerging field, a lot of research is focused on presenting evaluation research or solution proposals, followed by validation research. A lack of experience reports and opinion papers is also a clear indication of the emerging nature of the research. Figure 3 presents the approaches plotted against the number of papers with different challenges which gives a combined view of the selected studies and their distribution over these two dimensions. The size of the bubble represents the number of papers. It can be observed from the figure, that "communication" and "deployment" are well ahead of

Table IV
THE KEYWORDS ASSOCIATED WITH THE CHALLENGES IN THE LITERATURE

Challenges	Keywords	Mentions
Communication/Integration	API, REST, sockets, TCP, gateway, circuit breakers, load balancer, proxy, routing, router	29
Service Discovery	discovery, registration, service registry	11
Performance	QoS, performance, SLA	28
Fault Tolerance	fault, failure, recovery, tolerance, healing	23
Security	secure, authentication, authorization, OAuth, OAuth2, encryption, vulnerability, attack	13
Tracing and Logging	tracing, logging, debugging, profiling	8
Application Performance Monitoring	monitoring, application performance monitoring	24
Deployment Operations	operations, orchestration deployment, configuration, scaling, auto-scale, rolling upgrades, images, container	34

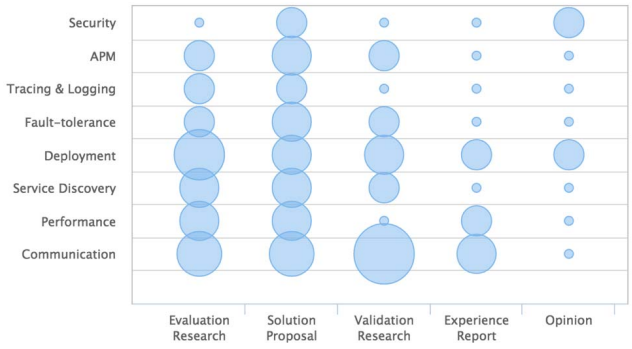

Figure 3. Research approaches against the number of papers with different challenges

the other challenges. It can also be noticed, that the "communication" and "deployment" challenges have more validation and evaluation papers.

The microservices architectural views/diagrams (RQ2):

Solution proposal and validation research types of papers were the main source to answer this question as they paid more attention to architectural modeling than other papers. In particular, the design and implementation sections of those papers provided figures of views/diagrams used along with their detailed explanations. However, although component/context diagrams were dominant in the literature, a wide variety of other graphical modeling views were also presented, although with no clear justification provided for the choice of a particular diagram. This lack of consistency in diagrammatic presentations may indicate a need to propose a comprehensive modeling view/language that best covers and describes a microservices-based architecture.

The graphical architectural views found in the literature were various and can be categorized into informal drawings with free boxes and lines, sets of UML diagrams each covering different aspects of the architecture, graphs with vertices and arrows and finally diagrams for SQL/NoSQL relational databases. Table V shows the diagrams used in the literature, their annotations and sets of papers that included each type of diagram. Interestingly, it has been noticed that there was no distinction between component diagrams and container diagrams in the literature. This implies that the trend of microservice architecture is to suggest placing one microservice, i.e. component, in one running environment, i.e. container, in order to achieve the ultimate independence and isolation of the microservices. In addition to the description diagrams covered earlier, description languages are also included in the literature to provide a more elaborated view of architecture details. Categories of different formats of description languages mentioned in the study included the following:

- Standard modeling languages, e.g. RAML and YAML.
- Specifically-designed modeling languages, e.g. CAMLE.
- Standard specification languages, e.g. Javascript (Node.js), JSON and Ruby.
- Specifically-designed specification languages, e.g. Jolie.
- Pseudocode for algorithms.

Quality attributes related to microservices in literature (RQ3):

To approach this question, a generalization of attribute names were necessary at first, since many alternative terms found in the papers indicated the same meaning of one attribute. Table VI shows each attribute and its alternative terms. It was noticed in the literature that well-known quality attributes of microservices architecture such as modularity, scalability, independence and maintainability were presented in almost all of the papers reviewed. However, some attributes scored fewer occurrences which implied lack of consideration. These attributes were basically security ID [15,18,23,32,33], load balancing ID [1,20] and organizational alignment ID [13,15]. In addition to the results of research questions 2 and 3 above, we decided to investigate a possible relationship between quality attributes in the literature and model views presented. For each quality attribute, we checked the modeling diagrams included in the same papers mentioning that attribute. This intersection method attempted to answer what type of modeling view is more suitable to demonstrate and/or test particular quality attributes in the architecture. More elaboration on our findings and insights derived from the results is provided in the next section.

IV. DISCUSSION

It can be observed from the results that microservices architectural style research is still in its infancy. Since the style is born out of industry, it has been noted that there are wide gaps between the current industry level and academia.

Table V
THE DIAGRAMS USED IN THE LITERATURE AND THEIR ANNOTATIONS

Diagram	Annotation	Paper ID numbers
Component/Container	Each micro-service is represented as a square/rectangle/oval and each line represents communication or data flows between components	1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32 and 33
Process/ Behavior	Each process of a microservice is a rounded square/rectangle and each arrow indicates an activity flow	1, 9, 11, 15, 22, 27 and 28
Sequence	UML diagrams	3, 5 and 18
Execution Timeline	Time grows from left to right on the x axis. Execution is represented as a rectangle parallel to the x-axis	4 and 9
Deployment	UML diagrams	7 and 32
Class	UML diagrams	20 and 30
Use Case	UML diagrams	28 and 33
Type Graph	Represents the needed resources and connection topology for each node (orchestrator) where cardinalities on edges represent the minimum and maximum number of allowed connections	3
Instance Graph	Represents the deployed service topology and components for each node (orchestrator)	3
Dependency Graph	Each node in the graph represents a microservice and the arrow indicates a dependency	4

Table VI
THE QUALITY ATTRIBUTES MENTIONED IN THE LITERATURE AND THEIR ALTERNATIVES

Attribute	Number of Papers	Alternative Terms and Expressions of Similar Meaning
Scalability	26	expandable, evolutionary
Independence	19	reducing complexity, isolation, loose coupling , decouple, distributed, containerization, autonomy
Maintainability	17	expandable, adaptability, changeability, flexible implementation, dynamically changing
Deployment	13	-
Health management	13	resilience, reliability, disaster recovery, no single point of failure
Modularity	13	single responsibility, reduce complexity, separate business concern, specialization, customizable
Manageability	12	self-managed, decentralized management, audibility
Performance	9	response times, transaction duration, throughput, efficiency
Reusability	7	pluggable
Technology heterogeneity	7	portability, freedom to choose a lot of technologies or programming languages
Agility	6	iterative, incremental, continuous delivery
security	5	-
Load balancing	2	workload intensity distribution
Organizational alignment	2	cross-functional team, reduce the conflict between developers and testers
Open interface	1	microservices should provide an open description of their APIs, GUIs and communication messages format

Most of the papers in our study were found to be either at a 'solution proposal' or a 'solution validation' stage, with validations based on lab-controlled experiments only. There are very few experience reports and opinion papers that can be found on the microservices architectural style. Moreover, microservices security is a very important challenge, which has not yet been very well researched. Even in industry, lots of service-based applications do not employ stringent security controls. It is also noted that tracing is one of the most common problems that is faced by all microservices style systems. Tracing a request through all the hops of business functions is a very difficult problem that demands attention from the academic community. Only a few prominent solutions are currently available in industry [48]. These solutions can help discover communication patterns, which can be used to discover dependencies between the services. A dependency graph helps architects in refactoring and making decisions with confidence.

As regards to RQ2, the literature presented different types of modeling diagrams and languages that describe aspects of microservices architecture as well as its lifecycle. Context and container/component diagrams with UML notations, for example, were extensively used to describe high-level static view of microservices architecture. To describe low-level design details, UML class diagrams accompanied with ERD data models, pseudocode for algorithms and additional textual description were used. UML use cases were used mainly to model validation and testing of microservices whereas UML sequence diagrams were used to sketch the communication between microservices. There was a particular kind of graph used to model deployment orchestration and automation called type graph/instance graph. Each type of graph represented the connection topology and needed resources to deploy a microservice whereas instance graphs represented each orchestrator service with its components. Interesting modeling languages presented in the literature were RAML, YAML and CAMLE. RAML and YAML (Swagger) are open standard modeling languages used to describe APIs of REST-

like messages needed for interacting and communicating with microservices. CAMLE is a specifically-designed conceptual graphical design for service-oriented systems that integrates with modeling language for agent-oriented systems called CAOPLE. According to the source paper, CAMLE/CAOPLE modeling method proved its efficiency in modeling the microservices architecture of CIDE, the proposed integrated development environment for building microservices systems. Code snippets of standard specification languages such as Javascript, JSON, Node.js and Ruby were used to describe the data model of messages communicated between microservices. A novel programming language called Jolie [35] was used to program and describe the architecture of its IDE which is also built using microervices.

Based on the previous analysis, it can be noted that modeling microservices with UML standard notations is comparable to creating another comprehensive modeling notation and also comparable to the use of informal drawings with free boxes and lines accompanied by a narrative. However, since a typical microservice based system consists of a number of containers and each container in turn contains one or more components, i.e. microservices, which in turn are implemented by one or more classes, then UML standard notation can provide a common set of abstractions and notation to describe microservices architecture. Therefore, using several UML diagrams, e.g. context, container, component, class, usecase, sequence, each showing a different part of the entire architecture will be effective to communicate software designs in an effective and an efficient way.

Results of RQ3 as in table VI have shown higher occurrences of, and hence more focus on, scalability, reusability, performance, fast agile development and maintainability. On the other hand, fewer occurrences, implying the need for future research, were related to security, load distribution (for multi cloud deployment with containers), continuous integration, organizational management and DevOps, as well as the automation of container management and deployment. Finally, having investigated the view model to quality attribute papers' overlap, the following findings have emerged:

- Papers concerning scalability, reusability, maintainability, manageability and deployment quality attributes also used component/container, class and deployment UML diagrams to demonstrate the potential of implementing those attributes.
- Use-case and sequence UML diagrams in addition to execution timelines assisted to compare and validate quality attributes of performance, deployment, security, maintainability and self-manageability of microservices architecture.
- Instance graphs/type graphs enabled the author of paper [3] to trace and validate quality attributes of health management, manageability and deployment automation.
- Dependency graphs co-occured with independence and maintainability quality attributes and also used to trace and test them.

The literature suggested many future trends as follows:

- Invent and automate approaches to empower the DevOps team so that development and operation functions are cooperative; hence, enabling the rapid and agile development and upgrade of applications, as well as deploying them on multiple platforms to meet customer needs.
- Investigate the impact of interrelationship between a process (service) and its context (situational factors) on microservice software process decisions.
- Allocate a specific programming language, e.g. Jolie, and IDE to develop microservices, e.g. CIDE.

V. Conclusions

This systematic mapping study has looked thoroughly into the available studies on microservices architecture and the relevant architectural challenges. The study uses two qualitative and quantitative synthesis methods, and addresses three key research questions. The first question addresses the architectural challenges that microservice systems face, where the researchers were able to explore all the published articles and studies that highlighted the gaps in microservices research and make suggestions about how to address some of the future solutions and initiatives. The second research question investigates which architectural diagrams and views, in addition to any methods or models, that are used to represent microservices architectures. Moreover, the last research question, states the possible quality attributes related to microservices that are presented in the literature. Our further work includes conducting a systematic literature review that takes into account other archiectural considerations of microservice architecture.

References

[1] Sascha Alpers, Christoph Becker, Andreas Oberweis, and Thomas Schuster. Microservice based tool support for business process modelling. In *2015 IEEE 19th International Enterprise Distributed Object Computing Workshop*, pages 71–78. IEEE, 2015.

[2] Marcelo Amaral, Jorda Polo, David Carrera, Iqbal Mohomed, Merve Unuvar, and Malgorzata Steinder. Performance evaluation of microservices architectures using containers. In *Network Computing and Applications (NCA), 2015 IEEE 14th International Symposium on*, pages 27–34. IEEE, 2015.

[3] Francois Andry, Richard Ridolfo, and John Huffman. Migrating healthcare applications to the cloud through containerization and service brokering. In *Proceedings of the International Conference on Health Informatics (BIOSTEC 2015)*, pages 164–171, 2015.

[4] Peter Bak, Roie Melamed, Dany Moshkovich, Yuval Nardi, Harold Ship, and Avi Yaeli. Location and context-based microservices for mobile and internet of things workloads. In *2015 IEEE International Conference on Mobile Services*, pages 1–8. IEEE, 2015.

[5] David Budgen, Mark Turner, Pearl Brereton, and Barbara Kitchenham. Using Mapping Studies in Software Engineering. In *Proceedings of PPIG 2008*, pages 195–204. Lancaster University, 2008.

[6] Phil Calçado. Building products at soundcloud—part iii: Microservices in scala and finagle. https://developers.soundcloud.com/blog/building-products-at-soundcloud-part-3-microservices-in-scala-and-finagle, June 2014.

[7] Augusto Ciuffoletti. Automated deployment of a microservice-based monitoring infrastructure. *Procedia Computer Science*, 68:163–172, 2015.

[8] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. Microservices: yesterday, today, and tomorrow. *arXiv preprint arXiv:1606.04036*, 2016.

[9] Adrian Fernandez, Emilio Insfran, and Silvia Abrahão. Usability evaluation methods for the web: A systematic mapping study. *Information and Software Technology*, 53(8):789–817, 2011.

[10] Luca Foppiano. A machine learning software for extracting information from scholarly documents. https://github.com/kermitt2/grobid, August 2016.

[11] Cristian Gadea, Mircea Trifan, Dan Ionescu, and Bogdan Ionescu. A reference architecture for real-time microservice api consumption. In *Proceedings of the 3rd Workshop on CrossCloud Infrastructures & Platforms*, page 2. ACM, 2016.

[12] Dong Guo, Wei Wang, Guosun Zeng, and Zerong Wei. Microservices architecture based cloudware deployment platform for service computing. In *2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, pages 358–363. IEEE, 2016.

[13] Sara Hassan and Rami Bahsoon. Microservices and their design tradeoffs: A self-adaptive roadmap. Technical Report CSR-16-01, University of Birmingham, School of Computer Science, April 2016.

[14] Baskaran Jambunathan and Y Kalpana. Multi cloud deploymentwith containers. *International Journal of Engineering and Technology (IJET)*, 8(1):421–428, 2016.

[15] Hui Kang, Michael Le, and Shu Tao. Container and microservice driven design for cloud infrastructure devops. In *2016 IEEE International Conference on Cloud Engineering (IC2E)*, pages 202–211. IEEE, 2016.

[16] kibana. Kibana analytics and search dashboard for elasticsearch). https://github.com/elastic/kibana, August 2016.

[17] Tom Killalea. The hidden dividends of microservices. *Queue*, 14(3):10, 2016.

[18] B. Kitchenham and S Charters. Guidelines for performing systematic literature reviews in software engineering, 2007.

[19] Holger Knoche. Sustaining runtime performance while incrementally modernizing transactional monolithic software towards microservices. In *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering*, pages 121–124. ACM, 2016.

[20] Alexandr Krylovskiy, Marco Jahn, and Edoardo Patti. Designing a smart city internet of things platform with microservice architecture. In *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, pages 25–30. IEEE, 2015.

[21] Vinh D Le, Melanie M Neff, Royal V Stewart, Richard Kelley, Eric Fritzinger, Sergiu M Dascalu, and Frederick C Harris. Microservice-based architecture for the nrdc. In *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pages 1659–1664. IEEE, 2015.

[22] James Lewis. Microservices: Adaptive systems for innovative organizations. https://www.youtube.com/watch?v=GDVcUM5wbxU, August 2015.

[23] James Lewis and Martin Fowler. Microservices. http://martinfowler.com/articles/microservices.html, March 2014.

[24] D. Liu, H. Zhu, C. Xu, I. Bayley, D. Lightfoot, M. Green, and P. Marshall. Cide: An integrated development environment for microservices. In *2016 IEEE International Conference on Services Computing (SCC)*, pages 808–812, June 2016.

[25] Divyanand Malavalli and Sivakumar Sathappan. Scalable microservice based architecture for enabling dmtf profiles. In *Network and Service Management (CNSM), 2015 11th International Conference on*, pages 428–432. IEEE, 2015.

[26] Suresh Marru, Marlon Pierce, Sudhakar Pamidighantam, and Chathuri Wimalasena. Apache airavata as a laboratory: architecture and case study for component-based gateway middleware. In *Proceedings of the 1st Workshop on The Science of Cyberinfrastructure: Research, Experience, Applications and Models*, pages 19–26. ACM, 2015.

[27] Karl Meinke and Peter Nycander. Learning-based testing of distributed microservice architectures: Correctness and fault injection. In *International Conference on Software Engineering and Formal Methods*, pages 3–10. Springer, 2015.

[28] Sam Newman. *Building Microservices*. " O'Reilly Media, Inc.", 2015.

[29] Petru Nicolaescu, Georgios Toubekis, and Ralf Klamma. A microservice approach for near real-time collaborative 3d objects annotation on the web. In *International Conference on Web-Based Learning*, pages 187–196. Springer, 2015.

[30] Rory V O'Connor, Peter Elger, and Paul M Clarke. Exploring the impact of situational context: a case study of a software development process for a microservices architecture. In *Proceedings of the International Workshop on Software and Systems Process*, pages 6–10. ACM, 2016.

[31] Claus Pahl and Pooyan Jamshidi. Microservices: A systematic mapping study. In *Proceedings of the 6th International Conference on Cloud Computing and Services Science*, pages 137–146, 2016.

[32] Srikanta Patanjali, Benjamin Truninger, Piyush Harsh, and Thomas Michael Bohnert. Cyclops: a micro service based approach for dynamic rating, charging & billing for cloud. In *Telecommunications (ConTEL), 2015 13th International Conference on*, pages 1–8. IEEE, 2015.

[33] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. Systematic mapping studies in software engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, EASE'08, pages 68–77, Swinton, UK, UK, 2008. British Computer Society.

[34] M. Rahman and J. Gao. A reusable automated acceptance testing architecture for microservices in behavior-driven development. In *Service-Oriented System Engineering (SOSE), 2015 IEEE Symposium on*, pages 321–325, March 2015.

[35] Larisa Safina, Manuel Mazzara, Fabrizio Montesi, and Victor Rivera. Data-driven workflows for microservices: Genericity in jolie. In *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, pages 430–437. IEEE, 2016.

[36] DI Savchenko, GI Radchenko, and O Taipale. Microservices validation: Mjolnirr platform case study. In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015 38th International Convention on*, pages 235–240. IEEE, 2015.

[37] Dmitry Savchenko and Gleb Radchenko. Microservices validation: Methodology and implementation. In *CEUR Workshop Proceedings. Vol. 1513: Proceedings of the 1st Ural Workshop on Parallel, Distributed, and Cloud Computing for Young Scientists (Ural-PDC 2015).—Yekaterinburg, 2015.*, 2015.

[38] Joe Stubbs, Walter Moreira, and Rion Dooley. Distributed systems of microservices using docker and serfnode. In *Science Gateways (IWSG), 2015 7th International Workshop on*, pages 34–39. IEEE, 2015.

[39] Yuqiong Sun, Susanta Nanda, and Trent Jaeger. Security-as-a-service for microservices-based cloud applications. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 50–57. IEEE, 2015.

[40] Johannes Thönes. Microservices. *IEEE Software*, 32(1):116–116, 2015.

[41] Giovanni Toffetti, Sandro Brunner, Martin Blöchlinger, Florian Dudouet, and Andrew Edmonds. An architecture for self-managing microservices. In *Proceedings of the 1st International Workshop on Automated Incident Management in Cloud*, pages 19–24. ACM, 2015.

[42] Sudhir Tonse. Microservices at netflix - challenges of scale. http://www.slideshare.net/stonse/microservices-at-netflix, August 2014.

[43] Matthias Vianden, Horst Lichter, and Andreas Steffens. Experience on a microservice-based reference architecture for measurement systems. In *2014 21st Asia-Pacific Software Engineering Conference*, volume 1, pages 183–190. IEEE, 2014.

[44] Nicolas Viennot, Mathias Lécuyer, Jonathan Bell, Roxana Geambasu, and Jason Nieh. Synapse: a microservices architecture for heterogeneous-database web applications. In *Proceedings of the Tenth European Conference on Computer Systems*, page 21. ACM, 2015.

[45] Mario Villamizar, Oscar Garcés, Harold Castro, Mauricio Verano, Lorena Salamanca, Rubby Casallas, and Santiago Gil. Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. In *Computing Colombian Conference (10CCC), 2015 10th*, pages 583–590. IEEE, 2015.

[46] Roel Wieringa, Neil Maiden, Nancy Mead, and Colette Rolland. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requirements Engineering*, 11(1):102–107, 2006.

[47] Herwig Zeiner, Michael Goller, Víctor Juan Expósito Jiménez, Florian Salmhofer, and Werner Haas. Secos: Web of things platform based on a microservices architecture and support of time-awareness. *e & i Elektrotechnik und Informationstechnik*, 133(3):158–162, 2016.

[48] Zipkin. zipkin.io. http://www.zipkin.io, August 2016.