XVII International Scientific Conference on Industrial Systems (IS'17)
Novi Sad, Serbia, October 4. – 6. 2017.
University of Novi Sad, Faculty of Technical Sciences,
Department for Industrial Engineering and Management
Available online at http://www.iim.ftn.uns.ac.rs/is17

Do the microservices improve the agility of software development teams?

Branislav Mišić
(University of Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia, bane.misic@outlook.com)
Milana Novković
(University of Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia, mnovkovic@uns.ac.rs)
Robert Ramač
(University of Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia, ramac.robert@uns.ac.rs)
Vladimir Mandić
(University of Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia, vladman@uns.ac.rs)

Abstract

Nowadays, agile approaches are becoming the mainstream paradigm for organizing the software development teams. However, in practice, some aspects of the agility of software development, e.g. the speed of development, scalability, reusability, and modularity are constrained or diminished due to the software product design itself or because of the inappropriate use of technologies. The recent advancement in the service-oriented architectures and cloud technologies resulted with a concept of microservices. Microservices are small independently releasable, upgradable and replaceable services that work together around a business domain. They support agile development principles and core aspects stated above. In this paper, we review the available literature for microservices and point out their benefits. The literature findings are used for designing a questionnaire for gathering information from an industrial project that utilizes the concept of the microservices. Overall results confirm microservice effectiveness on a variety of system elements and factors stated in literature. Therefore conclusion about question do the microservices improve the agility of software development teams could be simple yes. But on the other hand, there are quite a few statements to consider described in this paper if we want to answer how and to what extent do microservices improve the software development agility.

Key words: microservices, microservice architecture, microservices in practice, agile microservices

1. INTRODUCTION

In todays dynamic environment where the market is global and competition tends to evolve and adapt fast in order to stay on top, internet corporations have to be able to changes and introduce new features fast, which can be a real challange. The business organization needs to be able to respond to changes quickly, bring agility to its IT systems and infrastructure and at the same time maintain business stability. For a long time Service Oriented Architecture (SOA) was the best answer to organize huge applications around services and logically separate business domains. New technology and market changes force large companies to be more agile [8]. SOA as a forerunner of microservices represents a loosely-coupled architecture designed to meet the business needs of the organization [9]. Managing changes in case of large applications can be time-consuming, difficult to coordinate and control. Over time the main drawback of this architecture becomes visible. As more features are changed or added to the system, every feature has to be tested through the whole system to ensure compatibility. SOA system structure can be identified as a monolithic structure [9].

Monolithic architecture represents the architecture that consists of coarse-grained services and components that depend on each other. Monolithic deployment of a system represents a single point of failure, if the application fails for some reason, the whole set of services fails too [9]. Often used Enterprise Service Bus (ESB) based on application server, becomes the bottleneck that often generates high latencies providing a single point of failure [7]. Monolithic approach is good for small scale teams and projects. Meanwhile, in conditions where is important to achieve business competitiveness and enable scalability, flexibility and other requirements like fast development, short time to market and large team collaboration monolithic architecture starts being a huge barrier. The solution to overcoming that barrier was introduced as microservice architectural approach [10].

This paper tries to provide an answer to that question using qualitative research techniques on a real project. Using information gathered from extensive interviews with project team members, the research team was able to provide insight on the main question of this paper mentioned in the title.

The paper structure consists of the introduction to the main concerns of the paper in Section 1. Related work in Section 2 is filled with theory insight about microservices and software development agility. The methodology used for the research was detailed in Section 3. Results in Section 4, provide the detailed synthesis of information gathered by research interview. Section 5 consists of conclusion and discussion of the results from Section 4 regarding provided thesis in Section 2.

2. RELATED WORK

Microservices are defined to be contrasted to monolithic structure. Microservice architecture is an architecture based on several principles where related functionalities are combined and implemented as a single business capability represented as a microservice [3] [4]. This architecture represents an approach to developing applications by using independent services each running in its own process [5]. Microservices are small and focused and each microservice is built with its own source code, repository and delivery pipeline [12]. Microservices are defined as cohesive, independent processes interacting via messages, which represent modules of distributed application [4]. In terms of advantages, microservices make the system more efficient and highly available through the realization of each business capability, it's functionalities and related data as an independent service [3]. Loosely coupled services as the main trait of microservices enable independent, more frequent and rapid deployments that lead to faster development of new features and change of the existing ones [3,4,12]. Additionally, microservice architecture embraces high cohesion between this small operationally independent services [3,4]. Other advantages lie in their independence from other parts of the system so they can be deployed independently and therefore monitored and scaled separately [3,5]. Due to their way of packaging to containers, microservices are easy to relocate and replicate across heterogeneous platforms. Their ability to spread across geographic distances and data centers makes them highly available [3,4]. Microservices enable and support the implementation of ability to dynamically scale service instances according to the load there for enabling them to be elastic and have more fault tolerance in comparison to monolithic architecture. If one microservice fails others are not affected due to their isolation. Ability to be tested separately from the rest of the system enables isolation of the parts of the system that were changed or affected by the change [3]. Incorporating microservices into modern day application development can lead to additional benefits by the clear separation of concerns, loose coupling, and higher potential to adapt to changes thus leading to increased agility [6].

As agile software development is opposite of documentation-driven software development. Modern organizations are trying to bring agility to their IT systems by incorporating microservices. As a new software development style, agile development is focused on the talents and skills of individuals, modeling the process to specific people and teams [1]. Agile development theory is based on the agile manifesto, which was written by the practitioners who proposed many of the development methods. The manifesto states that agile development should focus on four core values: Individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation and responding to change over following a plan [2]. Agile software architecture could be described as a system that is built using a collection of small, loosely coupled components and/or services that unite together to satisfy the final end-goal. This architecture would style provide agility in many ways. Components could be created, modified, tested, replaced and easily maintained in isolation. This described model theoretically refers to microservices [13].

This new style of development also brings advantages as the contribution to the reduction of costs of moving information between people and reduction of time consumption in the process from the point of decision making to seeing the result of that decision [1]. In order to improve the ability to respond to changes organizations incline to make teams of people agiler through enforcing people factor in the software development process. In that process, organizations try to respect agile software development principles stated in [2].

Since microservices are modeled around specific business domain and are language neutral, development teams have the freedom to decide which programming languages to use regarding optimal technology for given tasks and team skills. If boundaries between business domains are well defined, microservices can be designed and sized efficiently, made in a way that they enable failure isolation. Microservices made this way enable developer with the right tooling to find out exactly which part of code caused the error since the code is decided properly according to specific business domain or functionality. Observability is also a very important aspect of every microservice based system. Tools that provide visual status of services enable quick response and provide the location of any problem, thus enabling the team to know the current state of every part of the system [12]. Furthermore, microservices allow companies to manage large applications using a methodology where incremental improvements are carried out by smaller teams on independent codebase and deployments. That considered, microservices bring some challenges of distributed systems and team development management practices that must be taken seriously [11]. Removing the overhead and risk of large-scale software development by using frequent iterations and smaller work increments, and prototyping as means of collaboration with users is one of the main principles of the Agile approach. Since the manifesto publication, the adoption of agile software development grew constantly as it provides just what the companies needed to deliver software fast and be more competitive on the market [13].

Brown in [14] claims that agile software development, continuous delivery, DevOps culture and microservices are all connected by a common set of goals. That is to be responsive as much as possible to customer needs while maintaining highest possible levels of software quality and system stability. Even if these architectural phases were developed in a specific order from the industry perspective, there’s no right sequence for an individual company to follow.

As McLarty said [13] "Microservices are the architectural phase of the agile progression." In practice, microservices downsides were not related to architectural pattern and implementation practices. Many companies struggle to apply microservice architecture pattern which requires time and experience for best results. Teams have to get reformed and change their habits in a way that benefits the new project and organization structure by Jan Stenberg [15]. Conway's law also called the mirroring hypothesis is acknowledged by reforming organization and team structure and, predicts that a development organization will inevitably design systems that mirror its organizational communication structure [16]. In his article, Jan Stenberg [15] describes five major factors of possible downsides and points of failure while using microservices which are not related to the architecture itself, but human factor, team and organizational project management in a potential agile environment and those factors are: Disagreement between developers, lack of developer experience, service boundaries causing barriers, code replication and service granularity planning. Table 1 consists of the theses from sources mentioned in this section. The theses represent base which is used to compare research results and they are covered in further sections.

Table 1. Main statements and benefits from the literature.

Thesis	Mark
In terms of advantages, microservices make the system more efficient and highly available through the realization of each business capability, it’s functionalities and related data as an independent service [3].	B1
Loosely coupled services as the main trait of microservices, they enable independent, more frequent and rapid deployments that lead to faster development of new features and change of the existing ones [3,4,12].	B2
Other advantages lie in their independence from other parts of the system so they can be deployed independently and therefore monitored and scaled separately [3,5].	B3
Due to their way of packaging to containers, microservices are easy to relocate and replicate across heterogeneous platforms. Their ability to spread across geographic distances and data centers makes them highly available [3,4].	B4
Microservices dynamically scale according to the load there for enabling them to be elastic and have more fault tolerance in comparison to monolithic architecture [3].	B5
Ability to be tested separately from the rest of the system enables isolation of the parts of the system that were changed or affected by the change [3].	B6
Incorporating microservices into modern day application development can lead to additional benefits by the clear separation of concerns, loose coupling, and higher potential to adapt to changes thus leading to increased agility [6].	T1
The manifesto states that agile development should focus on four core values: Individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation and responding to change over following a plan [2].	T2
Furthermore, microservices allow companies to manage large applications using a methodology where incremental improvements are carried out by smaller teams on independent codebase and deployments [1,11].	T3
Jan Stenberg describes factors of possible downsides and points of failure while using microservices which are not related to the architecture itself, but human factor, team and organizational project management: Disagreement between developers and lack of developer experience [15].	T4

3. METHODOLOGY

To answer the main question of this paper, the team decided to use qualitative research method. This method provides richer and more informative results by forcing the researcher to get into the core of the problem. It's based on answering the "why" question and helps with the questions that involve variables that are not easy to quantify. This method is described as less proof-oriented in comparison to quantitative, and is also more difficult to bring out the conclusion and summarize the results. The qualitative method, in this case, considers using hypotheses gathered in the literature review and carefully designing a semi-structured interview [17,18]. Semi-structured interviews represent a mixture of open-ended and specific questions, designed to not only get the information that is foreseen, but also unexpected types of information.

The interview consists of several groups of questions. Each group represents single hypotheses that will be covered from different points of view with multiple questions. The interview provided the collection of answers and information which is processed and summarized in three important concerns in Section 4. Those three concerns provide a deeper understanding of microservice architecture and agile development connections and how they affect each other on this particular industry project. One concerning microservices benefits, theory expectations and how they work in reality compared to other monolith structured architectures. Second concern describes microservices in terms of agile development, and tries to provide answers about the connection of agile development theory and microservices in practice. Third concern tries to describe how teamwork communication and human factor affects developing microservices and overall development agility.

Before the main questions in interview, demographic information was gathered in form of subject team role and years of experience in mentioned role. The main part of the interview covered 6 microservice benefits and 5 theses with a total of 17 questions. Interview provided information and overview, of how microservice architecture impacts a real project in terms of agility and inspect claims on this architecture benefits. Whole interview process per person was predicted to last a maximum of 45 minutes and it was recorded with the consent of the examinee.

4. RESULTS

The idea to use microservice architecture to make reusable service platform at TIAC d.o.o. emerged as the companies first microservice based project. The project was named Reqster. This platform represents end product, based on a set of unified reusable microservices that clients can use for their application needs without the need to build services from scratch. After setup phase was done and system running close to its full potential, with first clients, the project was a perfect candidate for this research. Each developer on the project was interviewed. There was a total of 5 interviewees, who intensively work on the project with various roles.

Table 2. Interview subjects by their role and experience

Subject no.	Position	Exp. (years)
S1	P1 (Project owner)	15
S2	P2 (System architect)	8
S3	P3 (Software developer)	2.5
S4	P3 (Software developer)	4
S5	P4 (User/Mobile developer)	3

Each project team role contributed to the understanding stated thesis about microservices and their agility they provide from a different perspective. At the start of the discussion, project roles are described with details and parts of the interview answers they contributed the most. Product owner role (S1) gave us more information about the idea of project and collaboration with clients. Interviewee (S1) provided concise, brief answers and valuable insight about how starting new microservice based project versus shaping existing monolith into one can be different. System architect and team lead role (S2) provided very detailed information about setting up the project and separating concerns. Some drawbacks and benefits were described with many details as discussed below. Leading the team, communication flow, and experience within team answers were also described. Software developers (S3,S4) focused more on details about the separation of concerns, testing, and overall communication as the important issues. Also, front-end developer (S4) gave a more detailed insight of how data manipulation on the client side is affected with microservices. As Reqster platform has to be connected to end user application by the development team or end user, client role emerged. Client role, mobile application developer (S5) provided mostly opinionated answers with hands on experience about how powerful and user-friendly it was, to use the microservice based platform to provide services for end clients and their applications. In what follows is a given summary of information gathered about the three main concernes mantioned in Section 3.

The first concern is specified using aggregated answers of seven questions described in further text. Independence and loose coupling, make the system easier to build. Microservices provide flexibility to deal with complicated tasks using a combination of many smaller services and components. In comparison to monolithic, microservices manage to work with only one working service, while others are in development and can still be productive. They scale easier than monolithic, easier to setup on more separate machines, they provide using different technologies and independence in a development of each individual or team. Deployment of monolithic application requires much more work. Microservices are more stable than monolithic structured architectures because components and parts of system don't rely on each other that much, or at all as stated by all interviewees. The system is much more resistant to errors, but on the other hand stated by (S5), for example, one user can make orders for another user in the current Reqster application. There is no much control over calls. Some nonexistant information can be called, all that has to be managed on the client side, stated (S5) from a client perspective. Monolith applications use less resources and often get everything they need in a single call was opinion of (S4) as front-end developer. In the case of microservices where the request has to travel through the whole application to serve some complicated request, multiple service calls have to happen to complete single data set. Microservices use way less resources if simple calls are needed. Eventually all depends on system size. Microservices are somewhat more iterative because smaller components are easier to manage. Changes don't lead to much or any changes to the rest of the system. On the other side if microservices require change on one or two services, they act faster than other architectures, but in the case of changes on the whole set of services, monolithic are considered to be easier. On front-end side, microservices could be a little bit complicated to use stated by (S4), because they require more separate calls in order to complete one specific task. Separate calls lead to many asynchronous calls, which may lead to writing more code and slower response time. Initially, microservices based system is harder to setup, but later when the system gets mature, it's much easier to maintain and acts faster to changes than monolithic systems. It depends on how fast can it reach full operability, due to the complexity of the system. Opinionated answers conclude that scaling microservice based system would be much easier due to the ability to spin more microservices that need scaling on more machines or containers, at any location. Testing is equally important at microservices as on other architectures, but in the case where clients can have more versions of the same service, testing becomes very important. Each microservice represent smaller code base and overall fewer functionalities, which leads to covering cases easier with tests. Also testing microservices requires more planning ahead and being cautious while testing cross component communications. Tests can be often reused and easily multiplied. Testing can be more complicated in bigger systems but overall it is easier for medium and smaller applications when compared to monolith structured applications, as (S2) had different opinion. Monitoring is also very important for microservices in comparison with other architectures, because the system can function even if some microservice doesn't. Being able to detect which one, how and why it stopped working makes microservice system considerably easy to maintain. Setup can easily go wrong because a lot of decisions must be made in terms to make separation of concerns.

In the second concern, microservices were investigated in terms of compatibility with agile development theory and agile manifesto. Most answers were opinionated regarding the experience on the project. Teamwork benefits are more visible at microservice architecture because of cleaner separation inside the team. When the system is fully operational, it's easier to organize work of one microservice team. Microservices support claim that individuals and their interaction are more important than processes and tools. They support different technologies for every single microservice to be used. On the other side sole mentioned only by the (S2), there are some conventions and rules when comes to developing microservices, but overall they support teamwork more than other architectures because of their flexibility. Documentation importance is more in focus when it comes to microservice architecture. It’s equally important to have software that works, and development in focus same as detailed documentation about service APIs. Documentation of business logic is usually less extensive, and less in focus, while API description and how it works is greatly more detailed, updated and used in development. In the case of Reqster project, the team is more prone to having close collaboration with clients than having extensive contracts. They require some basic contract to be made at the beginning, but constant collaboration is often required during development and later in the maintenance of the system, opinionated by the development team (S2,S3,S4) and confirmed by product owner (S1). Microservices made easy responding to often changes over following a strict plan that’s hard to change, which is the main trait of the agile development as stated by all examinees.

Human factor while developing microservice based applications is described as the third concern that interview answers covered. In terms of time put in discussions and meetings, microservices require about the same time as classic, monolith structured architectures. Depending on team role, inexperienced team members need less effort to start working with microservices than with other monolith structured architectures stated by development team (S2,S3,S4). Basic theory, rules, and conventions about microservices are required to be able to understand and work with them, and they are well documented and available on the internet. Separated concerns make new team members quickly familiar with their tasks and overall system. Discussions while developing microservice based application take place often, but shorter in length and usually have less importance. In terms of quantity, they take the same amount of time than any other architecture. Having frequent discussions between team members and other teams prevents some major misunderstanding and problems to occur as (S3,S5) stated.

5. CONCLUSION AND DISCUSSION

On global informational technology market, microservices gain more on popularity by the day. Having fast, reliable, and overall agile system that is able to respond to changes in no time, certainly became a priority. Yet implementing microservice architecture can become a challenge in many ways, so the real benefits and overall agility of using microservices in practice become questionable. Information gathered on microservice benefits and comparison with other architectures, confirm stated thesis (B1,B2,B3) in table 1 Section 2. As all the examinees state that Independence and loose coupling, make the system easier to build, they also state that microservices provide flexibility to deal with complicated tasks. Monitoring and log tracking were also stated as very important for microservices in comparison with other architectures because it represents the only and true way of tracking the overall system state. On theses (B4,B5) opinionated answers state that microservices scale easier than monolithic, and are easier to setup on more separate machines. Scaling microservice based system would be much easier due to the ability to spin more microservices that need scaling on more machines or containers, at any location. Thesis (B6) is partially proven and is covered by states that testing cross component services require more ahead planing and can be more complicated in bigger systems. Also, test can be often reused and easily multiplied. Testing logically smaller components is a lot easier than on other architectures. On (T1, T3) results state that if microservices require change on one or two services, they act faster than other architectures, but in the case of changes on the whole set of services, monolithic are considered to be easier. Also, results state that microservice based system is much easier to maintain and acts faster to changes than monolithic systems. In thesis (T2) agile manifesto was aligned with microservices in three out of four statements, strongly. Only the statement “working software over comprehensive documentation” was not completely supported by states that It’s equally important to have software that works, and development in focus same as detailed documentation about service APIs. Also, results state that documentation of business logic is usually less extensive, and less in focus, while API description and how it works is greatly more detailed, updated and used in development. About human factor theses (T3,T4) results state that in terms of time put in discussions and meetings microservices require about the same time as any other architecture. Inexperienced team members need less effort to start working with microservices than with other architectures because separated concerns make new team members quickly familiar with their tasks and overall system. Discussions take place more often, but shorter in length and usually have less relevance. Results also state that having frequent discussions between team members prevents misunderstanding and communication-related problems to occur.

Overall results confirm microservice effectiveness on a variety of system elements and factors stated by the thesis. Therefore conclusion about question do the microservices improve the agility of software development teams could be simple yes. But on the other hand, there are quite a few statements to consider described in this paper if we want to answer how and to what extent do microservices improve the software development agility.

6. REFERENCES

[1] Cockburn, Alistair, and Jim Highsmith. (2001): 131-133.

[2] "Agile software development, the people factor." Computer 34.11 Fowler, Martin, and Jim Highsmith. "The agile manifesto." Software Development 9.8 (2001): 28-35.

[3] Dragoni, Nicola, et al. (2017) "Microservices: How to make your application scale." arXiv preprint arXiv:1702.07149

[4] Dragoni, Nicola, et al. (2016) "Microservices: yesterday, today, and tomorrow." arXiv preprint arXiv:1606.04036

[5] Namiot, Dmitry, and Manfred Sneps-Sneppe. (2014) "On micro-services architecture." International Journal of Open Information Technologies 2.9.

[6] Di Francesco, Paolo, Ivano Malavolta, and Patricia Lago. 2017 "Research on Architecting Microservices:Trends, Focus, and Potential for Industrial Adoption." Software Architecture (ICSA), 2017 IEEE International Conference on . IEEE, 2017.

[7] Mahmood, Zaigham. (2007): 74-78. "The promise and limitations of service oriented architecture." International journal of Computers 1.3

[8] Yu, Yale, Haydn Silveira, and Max Sundaram. 2016. "A microservice based reference architecture model in the context of enterprise architecture." Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), 2016 IEEE . IEEE, 2016.

[9] Fowler, Martin, and James Lewis. (2014) "Microservices." ThoughtWorks. http://martinfowler.com/articles/microservices.html [last accessed on June 21, 2017].

[10] Jaramillo, David, Duy V. Nguyen, and Robert Smart. 2016 "Leveraging microservices architecture by using Docker technology." SoutheastCon, 2016 . IEEE.

[11] Villamizar, Mario, et al. 2015 "Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud." Computing Colombian Conference (10CCC), 2015 10th . IEEE.

[12] Jaramillo, David, Duy V. Nguyen, and Robert Smart. 2016 "Leveraging microservices architecture by using Docker technology." SoutheastCon, 2016 . IEEE.

[13] Matt McLarty, (2016). “Microservice architecture is agile software architecture“ http://www.infoworld.com/article/3075880/application-development/microservice-architecture-is-agilen software-architecture.html *[last accessed on June 22, 2017]*.

[14] Simon Brown. (2013) "What is agile software architecture?" http://www.codingthearchitecture.com/2013/09/03/what_is_agile_software_architecture.html *[last accessed on June 22, 2017]*.

[15] Jan Stenberg. (2014) “Experiences from Failing with Microservices“ https://www.infoq.com/news/2014/08/failing-microservices *[last accessed on June 23, 2017]*.

[16] Kwan, Irwin, Marcelo Cataldo, and Daniela Damian. (2012): 90-93. "Conway's law revisited: The evidence for a task-based perspective." *IEEE software* 29.1.

[17] Shull, Forrest, Janice Singer, and Dag IK Sjøberg, eds. 2007 *Guide to advanced empirical software engineering*. Springer Science & Business Media.

[18] Creswell, John W. 2013. *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications.