Designing Online Healthcare Using DDD in Microservices Architecture

M Rizki, A N Fajar* and A Retnowardhani

1 Information Systems Management Department, BINUS Graduate Program - Master of Information Systems Management, Bina Nusantara University Jakarta, Indonesia 11480

* afajar@binus.edu

Abstract. Today, the system development had been transformed to service-oriented architecture. It more specific in microservices architecture to responds the agility and flexibility in software maintenance. This transformation is an important part that promoting independence, speed and security, while still kept balanced with the company's condition through comprehensive and effective steps. Culture of the development cycle of a company highly affected by the adaptation of microservices architecture. Thus, in order to achieve flexibility, speed and security, which are the main goal of adapting microservices, changes must be deliver well and sustainably. This study focused on designing of microservices architecture for health system using domain driven design (DDD) approach.

1. Introduction

The rapid development of various technology offered products online resulted in many more companies engaged in online products and services business. This resulted in these companies required quick and appropriate respond to meet the market demands, to present attractive and responsive features and move quickly in response to problems. However, all of that must be delivered in harmony with careful and sustainable architectural planning [1]. Difficulty in changing the systems is more difficult for dynamic condition in modifying and customizing. It is still possible to maintain and develop monolithic software with such conditions, but eventually it becomes clear that architectural changes throughout the application must be made [2]. There are several things needed to take attention, such as to change a feature in a code that is already large. It is also complex which involves very close functional dependency in it. The current software system migration to microservice bring many benefits including the ability to improve availability and scalability for various parts of the system. It can accommodate the ability to use different technologies and vendors. It can also can reduce time for release and completing a better code base [3]. These benefits of this study is to make a recommendation the process of transformation systems from monolithic architecture systems to microservices architecture (MSA). It will provide stages that can be carried out for the designing of microservices architecture (MSA). Besides that, it can contribute to further research on migration patterns that can be done at an online health care service company. As we know, healthcare is a domain that consists of large information exchange and have many entities in the ecosystem. In order to accommodate the ecosystem, online healthcare is part of systems. In this study, monolithic architecture has already implemented. However, in this condition, it is difficult to maintain and manage the systems while the changes are fast and
dynamic. In order to make the software system maintenance is easier, microservices architecture can be implemented. Besides that, we used domain driven design (DDD) to design microservices architecture.

2. Related Research
Researchers examine the evolution and exploration the challenge for the future of microservices [4]. According to [5] that had been conducted a study of microservices feasibility of economics and budget in the implementation of microservices, involving simple ways to change components by reducing the tendencies that can cause or add new problems throughout the system. A detailed study of microservices point of view with members of industry working groups provide clear and specific definitions of MSA, core principles of MSA, and also provide a comparison of MSA with Service-or-embedded-architecture (SOA) [6]. Moreover, a research about the importance for doing a decision-making process using microservices highlight the importance of understanding context and the importance of choosing an appropriate life cycle model for the project scope, greatness of impact, complexity, changing necessities and chances. The general domain of situational factors that affect in microservices development process can be seen as fundamentally essential for the future of software development, classification into various things [7].

3. Methodology
The research stages consists of analysis of monolithic architecture, analysis the systems requirement, and designing MSA. This research begins with the analysis of business processes in online health care. In this study, we use sample business processes in online healthcare. Related to designing MSA, we proposed SOA-based application design using MSA. We domain-driven design (DDD) approach by [8]. Data collection is required to be held at the beginning of preparation for this stage by conducting field research methods, namely visits and interviews with relevant parties. Another study has been proposed by [9] which is related to microservices design.

4. Result and Discussion
When we develop a software or system, the first we must understand things related with software or system domain. For this case, healthcare is a domain, and online healthcare is the systems that has been implemented. Good understanding of the health domain; in terms of users (patients), facilitators (hospitals and doctors) and finance (insurance) are needed. After we doing the analysis the current condition, we found that the legacy system architecture uses monolith application for all of these services. There are much Functional systems are still interrelated with one another in a large codebase. Therefore, it is necessary to review the architecture that can provide more benefits for the company in software development iteration, both of architecture and harmonization between departments. Microservices ease this process by utilize the Agile Development approach in software development lifecycle. It can support conceptual framework used for making responsive software development efforts. Besides that, the changes in the early phases for functional validation, continuous improvement and encouraging fast and flexible responses to changes in development process. The optimum scheme to apply this is by treating software as a reflection of the domain. It needs to incorporate core concepts and domain elements, also precisely realize the relationships between them. It is essential to understand the existing infrastructure and data information flow when a company has chosen to carry out large scale architecture transformation. Figure 1 below shown the activities of booking journey in online healthcare:
Figure 1. Booking Journey.

Figure 1 above shows the flow processes in booking journey from start to finish. The next stage is analysis of Insurance Journey that can be shown in figure 2 below:

Figure 2. Insurance Journey.
After the analysis business processes in booking journey and insurance journey, we continued to analyze the endpoints current condition. Identification of existing endpoints can help in mapping the problem for the conversion of monolith architecture to microservice. From the collection of endpoints, it can be further analyzed to make classification based on the functional and domains that interact with these endpoints. Various important endpoints are obtained for further analysis in order to get a big picture of the system about how some products interact with each other. Following we will present a list of endpoints that are used in serving users to interact with the system, as those endpoints are the APIs used by applications that interact directly with the system backend with additional API versioning for future needs. The list of endpoints above is part of the existing APIs included in the material for discussion with domain experts and developers to trace the domains that interact with these endpoints. After all the required endpoints have been collected, the discussion continues on ownership or ownership of endpoints regarding who has the duty to create, maintain and make modifications to certain endpoints. In this case, the company divides the team into three parts namely Chat, Booking and Content User teams. Endpoint ownership is an important determinant when there are problems and modifications to the API. The list of teams responsible for endpoints is known, the next step is to classify endpoints based on functional domains that can represent all the services available. This can be divided into several services that complement each other, in groups that are increasingly focused on the similar specific functionalities. The endpoints divided into 10 services include Common Services, Content Services, Medical Record Services, Campaign Services, User Services, Chat Services, Doctor Chat Services, Doctor Booking Services, Booking Services, and Insurance Services. Single application development will be conducted as a series of small services that are part of microservices. The results of this study consist several models and designs of proposed software and can be analyzed through information contained in the existing functional documents, architecture, databases and interviews with related parties. We used the software stack in figure 3 below:

| No | Category                                | Candidate Name               |
|----|-----------------------------------------|------------------------------|
| 1  | Language                                | Ruby                         |
| 2  | Database                                | MongoDB                      |
| 3  | Cache                                   | Redis                        |
| 4  | Web Server                              | Fusion Passenger             |
| 5  | API Gateway                             | Custom Built                 |
| 6  | Monitoring Logs                         | Fluentd                      |
| 7  | Monitoring Dashboard                    | Grafana                      |
| 8  | Search Database                         | Elastic Search               |
| 9  | Web UI                                  | Polymer UI                   |
| 10 | Web Framework                           | Ruby on Rails                |
| 11 | Container Management/Orchestration      | Kubernetes                   |
| 12 | HTTP Proxy                              | HAProxy                      |
| 13 | Container                               | Docker                       |
| 14 | Services Registration                   | Zookeeper                    |
| 15 | Sync Communication                      | HTTP                         |
| 16 | Async Communication/Message Queue       | RabbitMQ                     |
| 17 | Security Token                          | JWT                          |
| 18 | Circuit Breaker                         | Hystrix                      |
| 19 | Test Framework                          | TDD, Rspec                   |
| 20 | CI/CD                                   | Bitbucket/Jenkins            |

*Figure 3. Software Stack.*
The software stack presented can be implemented with the function of a distributed architecture that is presented in a literature review, especially Software-Oriented Architecture.

5. Conclusion
Restructuring from existing code in production environment which already go live can be a risky business. Thus, efforts in increasing success chance of restructuring and good test coverage are needed. Adopting microservice architecture requires organizational change. Therefore, adaptability of the organization from the old to the new architecture is essential. The service ownership rights of each team that has responsibility for the service need to be resolved and decided. It means that the ownership will fall to those team who develop, test, install, and maintain services up to the production level.

Acknowledgment
The authors acknowledge contribution of the other research team members and Bina Nusantara University. This research was funded by Research Grant from DIKTI in 2020 with the entitle “Online Health Care Based on Microservices Architecture”.

References
[1] G. Gregory Mihuba, M. Joseph Shundi, B. Obadia Kyetuza, M. Anthony Msafiri. (2019). Design of Telemedicine System Based on Mobile Terminal. International Journal of Emerging Trends in Engineering Research, Vol. 7, no. 1, pp. 5-6, Jan 2019.
[2] Kalske, M., Mäkitalo, N., & Mikkonen, T. (2018). Challenges When Moving from Monolith to Microservice Architecture. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). https://doi.org/10.1007/978-3-319-74433-9_3.
[3] Balalaie, A., Heydarnoori, A., Jamshidi, P., Tamburri, D. A., & Lynn, T. (2018). Microservices migration patterns. Software - Practice and Experience. https://doi.org/10.1002/spe.2608.
[4] Jamshidi, P., Pahl, C., Mendonca, N. C., Lewis, J., & Tilkov, S. (2018). Microservices: The journey so far and challenges ahead. IEEE Software. https://doi.org/10.1109/MS.2018.2141039.
[5] Singleton, A. (2016). The Economics of Microservices. IEEE Cloud Computing. https://doi.org/10.1109/MCC.2016.109.
[6] Christudas, B., & Christudas, B. (2019). Microservices Architecture. In Practical Microservices Architectural Patterns. https://doi.org/10.1007/978-1-4842-4501-9_4.
[7] O’Connor, R. V., Elger, P., & Clarke, P. M. (2016). Exploring the impact of situational context - A case study of a software development process for a microservices architecture. Proceedings - International Conference on Software and System Process, ICSSP 2016. https://doi.org/10.1145/2904354.2904368.
[8] Evans, E. (2002). Tackling Complexity in the Heart of Business Software. Pattern Languages of Program Fajar, A.N., Novianti,E, Firmansyah.T. Desing and Implementation of Microservices System Based on Domain Driven Design. International Journal of Emerging Trends in Engineering Research, Vol.8.No.