Partitioning Microservices: A domain Engineering Approach

Abstract—**In this paper, we attempt to define a micro service and show how micro services can be partitioned based on a domain driven engineering approach.**

I. Introduction

Microservices is the new buzz word around software architecture patterns today. Microservices provide several advantages over monolithic systems. Some of these include the ability to make rapid functional changes which contributes to achieving high integrity factors such as maintainability and scalability, continuous software delivery, and delivering software into production [1].

Add operational definition for monolithic system, and micro service, generally, for scope of paper, some relationship to sofware architecture patterns, rationale

However, a major question and challenge is on how to introduce microservices, and arrive at an appropriate size for a microservice [2], [3]. A major question to be answered is where component boundaries should lie [3].

Some suggestions have been proposed to this effect, including among others, aspects on if the microservice will be a user service, and therefore a decision being made based on the tooling (with leaning towards the usage of lightweight tools), size being determined by the number of lines of code (with recommendations of not exceeding a couple thousand lines of code), and functionality in terms of the microservice accomplishing specifically only one task [1]. [2] proposes partitioning services by use case. Other strategies are to partition by verbs, nouns, or resources, and the scaling cube [4]. According to [5], independent services should focus service boundaries on business boundaries, so as to avoid the difficulties introduced when the service becomes too large. He also postulates that a micro service as something that could be rewritten in two weeks, with proper alignment to team structures.

Size for a microservice is important, because the smaller the service, the more the benefits of microservice architecture are maximized [5]. However, there is a lot of ambiguity around the right size of a micro-service, and there is lack of good guidelines for designing a micro-service in terms of scope or size. The challenges that arise encompass how a micro-service can be partitioned in the right size to ensure loose coupling, such that the service can easily be changed to keep up with business and technical demands [...**reference**...].

According to [6], three keys to successful microservices are componentization, collaboration, and reliable connections and controls. That not withstanding, the microservice approach must contend with some issues such as integration between communicating applications [1], and complexities that arise from creating a distributed system. These include testing, deployment and increased memory consumption [2]. [3] cites the need for microservices to design for failure by possibly, automatically restoring the failed service.

Domain Driven Design (DDD) provides a number of useful patterns for dealing with the kind of complexity encountered in designing distributed systems and with large and complex domains, by breaking the domain into a series of bounded contexts.

II. Related Work

Something on Microservices building blocks (Joselyne)

A micro-service is small and focused on doing one thing. Decomposing an application to a micro-service is a crucial task, with some developers relating the size of a micro-service to the number of lines of code (LOC). They recommend that a service should not exceed(10 to 100 LOC) [7]. However, this practice is not genuine as micro-services are built using different technology stacks, which differ on LOC. Other strategies partition a micro-service as one kind of development and deployment unit. However, in literature, micro-services are built based on the need to address one business capability, or one business functionality at a time. Concepts, can be composed with different business capabilities at different levels. [...**References**...]

Business capability: Business capability defines what a system does in enabling the organization's capacity to successfully perform a unique business activity [8]. In agile development, it is an important way of combining data that have something in common such as functionality, rather than using collections of data entities that expose CRUD-style methods. Because it helps to understand demand impact on application architecture at an early stage. Developers still have a problem on using business capability as boundary of micro-service, on defining which level of functionality a micro-service should fit, so that it can't be too small a service that depends on other services, leading to a decrease in its autonomy or too big to lose the dependency.

Something on Domain Engineering building blocks (Doreen)) What in domain engineering helps to breakdown into services, can be used to breakdown?

III. PROCEDURE

Our approach is concerned with partitioning a micro-service following domain driven engineering principles by considering asynchronous and synchronous communication.

REFERENCES

[1] J. Thönes, "Microservices," *IEEE Software*, vol. 32, no. 1, pp. 116–116, 2015.

[2] D. Namiot and M. Sneps-Sneppe, "On micro-services architecture," *International Journal of Open Information Technologies*, vol. 2, no. 9, 2014.

[3] M. Fowler and J. Lewis, "Microservices," *Viittattu*, vol. 28, p. 2015, 2014.

[4] M. L. Abbott and M. T. Fisher, *The art of scalability: Scalable web architecture, processes, and organizations for the modern enterprise*. Pearson Education, 2009.

[5] S. Newman, *Building Microservices*. " O'Reilly Media, Inc.", 2015.

[6] O. Garrett, "Three keys to successful microservices," http://www.infoworld.com/article/2936148/application-development/three-keys-to-successful-microservices.html.

[7] G. Schermann, J. Cito, and P. Leitner, "All the services large and micro: Revisiting industrial practice in services computing," in *International Conference on Service-Oriented Computing*. Springer, 2015, pp. 36–47.

[8] U. Kalex, "Business capability management: Your key to the business board room," http://www.opengroup.org/johannesburg2011/Ulrich%20Kalex%20-%20Business%20Capability%20Management.pdf.