Domain Engineering for Weather Information Services

Doreen Tuheirwe-Mukasa
dtuheirwe@cis.mak.ac.ug

Makerere University

April 2017

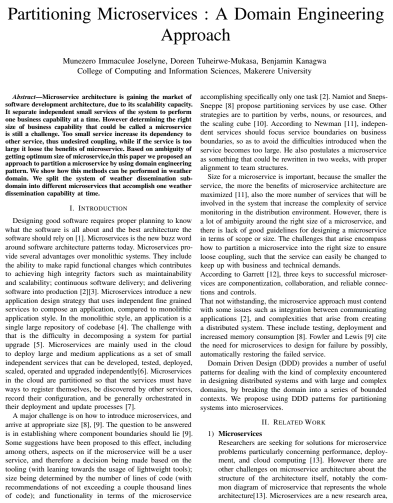
Domain Analysis

Last Month

- Engage UNMA - atmost 5 identified personnel for guided questionnaires
- Publication for domain model
- Refine initial Architecture with details from exploration - **pending**

Analysis Objective

- 10 questionnaires filled and received from UNMA personnel
- Publication submitted to IEEE Africon 2017 Conference http://africon2017.org/ - 17th April
    - Present weather domain, sub-domains, identify components, relationships and operations (presented as natural language concepts)



Partitioning Microservices : A Domain Engineering Approach

Munezero Immaculee Joselyne, Doreen Tuheirwe-Mukasa, Benjamin Kanagwa
College of Computing and Information Sciences, Makerere University

Abstract—**Microservice architecture is gaining the market of software development architecture, due to its scalability capacity. It separate independent small services of the system to perform one business capability at a time. However determining the right size of business capability that could be called a microservice is still a challenge. Too small service increase its dependency to other service, thus undesired coupling, while if the service is too large it loose the benefits of microservice. Based on ambiguity of getting optimum size of microservice,in this paper we proposed an approach to partition a microservice by using domain engineering pattern. We show how this methods can be performed in weather domain. We split the system of weather dissemination sub-domain into different microservices that accomplish one weather dissemination capability at time.**

I. Introduction

Designing good software requires proper planning to know what the software is all about and the best architecture the software should rely on [1]. Microservices is the new buzz word around software architecture patterns today. Microservices provide several advantages over monolithic systems. They 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 [2][3]. Microservices introduce a new application design strategy that uses independent fine grained services to compose an application, compared to monolithic application style. In the monolithic style, an application is a single large repository of codebase [4]. The challenge with that is the difficulty in decomposing a system for partial upgrade [5]. Microservices are mainly used in the cloud to deploy large and medium applications as a set of small independent services that can be developed, tested, deployed, scaled, operated and upgraded independently[6]. Microservices in the cloud are partitioned so that the services must have ways to register themselves, be discovered by other services, record their configuration, and be generally orchestrated in their deployment and update processes [7].

A major challenge is on how to introduce microservices, and arrive at appropriate size [8], [9]. The question to be answered is in establishing where component boundaries should lie [9]. 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 [2]. Namiot and Sneps-Sneppe [8] propose partitioning services by use case. Other strategies are to partition by verbs, nouns, or resources, and the scaling cube [10]. According to Newman [11], 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 a microservice 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 [11], also the more number of services that will be involved in the system that increase the complexity of service monitoring in the distribution environment. However, there is a lot of ambiguity around the right size of a microservice, and there is lack of good guidelines for designing a microservice in terms of scope or size. The challenges that arise encompass how to partition a microservice into the right size to ensure loose coupling, such that the service can easily be changed to keep up with business and technical demands.
According to Garrett [12], 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 [2], and complexities that arise from creating a distributed system. These include testing, deployment and increased memory consumption [8]. Fowler and Lewis [9] cite 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. We propose using DDD patterns for partitioning systems into microservices.

II. Related Work

1) **Microservices**
   Researchers are seeking for solutions for microservice problems particularly concerning performance, deployment, and cloud computing [13]. However there are other challenges on microservice architecture about the structure of the architecture itself, notably the common diagram of microservice that represents the whole architecture[13]. Microservices are a new research area,

Next Steps - Design Objective

- Analyse UNMA feedback, refine model with details from exploration
- Publication for domain model
    - Target International Journal on Software and Systems Modeling (SoSyM) http://www.sosym.org/

Thank you! Comments, suggestions, questions, reactions?