Modeling Microservices with DDD

Paulo Merson
Brazilian Federal Court of Accounts (TCU)
Brasília, DF – Brazil
pmerson@acm.org

Joseph Yoder
The Refactory
Urbana, IL – USA
joe@refactory.com

***Abstract*— Many have suggested using Domain-Driven Design (DDD) to help define the functional scope of microservices. But how to apply this idea in practice is not clear to everyone. DDD is a domain modeling technique created in the early 2000s. Microservices is an architecture style that became popular in 2015 as means to break software solutions into a set of independently deployed services. In this full-day tutorial we'll cover basic DDD concepts and discuss why and how DDD can help to create microservices with better availability, scalability, reliability, and modifiability. Using examples, we'll navigate from a domain model created using DDD to the design of both synchronous (REST-based) and asynchronous (reactive) microservices. We'll explore five different microservice design scenarios around DDD aggregates, bounded contexts (BC), domain events and other strategies for inter-BC interaction.**

Keywords—domain-driven design (DDD), microservice

I. The Topic, State of the Practice, and Relevance

Domain-Driven Design (DDD) saw significant attention from the software development community since it was proposed by Eric Evans in 2003 [1]. Many organizations applied DDD (or parts of it) successfully and more recent publications have given us practical advice and best practices on using DDD [2][3][4].

Microservice is an architecture style where the applications comprise a suite of small, independently deployed backend services, each running in its own process or VM, and typically interacting via http APIs or asynchronous messaging [5]. The defining design constraint of the microservice style is that the deployment unit shall contain only one service or a small set of cohesive services [6].

Microservices became popular in 2014-2015 to the point that it was then a buzzword. Since then, we as an industry identified one of the main challenges of microservice design: creating microservices that are not too large or too small, and hence contain the right amount of functionality. Many authors saw in DDD a modeling approach that could help to overcome this challenge. Sam Newman says that the modular boundary established by a DDD bounded context (BC) becomes an excellent candidate for a microservice [7]. In recent conference presentations, Eric Evans himself has talked about microservices as context boundary, but he also says there are different types of BCs that map to microservices [8][9]. In his recent book, Chris Richardson describes the relation between DDD and microservices in the *Decompose by Subdomain* pattern [10]. He goes on to discuss both BC and aggregate as DDD elements that can provide functional boundaries for microservices. Neal Ford and colleagues say that BCss serve as a quantum boundary in a microservices architecture [11].

Thus, with the popularity of the microservice architecture style, DDD has seen rekindled interest from the software community. However, much of the conference talks, blog posts and even book sections discussing the relation between DDD and microservice design fall short at providing a concrete design solution. One is left wondering, for example, how does microservice *Purchase Order* proposed in this box-and-line diagram translate to real implementation technologies? How do these boxes map to actual REST controller modules, Kafka listeners, ETL batch processes, etc.? What design elements can become docker images?

In this tutorial we take a step further to discuss and describe *how* a domain model created using DDD terminology can be translated into a microservice-based design. To describe a realistic design solution, we use different technologies, such as Spring Boot, Kafka, and Docker—the proposed microservice design uses types of components and connectors provided by these technologies. The *examples* of technologies used in the tutorial allow participants to relate or correlate the design solutions to their technologies of choice. However, the guidelines for DDD modeling to microservice design in the tutorial are platform agnostic.

The relevance of this tutorial is linked to the relevance of the microservice architecture style, since we address the challenge that is central to successfully applying the style: defining the functional scope of microservices. As many have proposed, we employ DDD as the technique to understand the problem space, but we shed light on the microservice-based solution space to allow a closer look at components and connectors created with today's technologies.

II. Target Audience and Prerequisites

This tutorial is primarily targeted at practitioners that work with distributed systems in general, or microservices in particular. Educators who teach distributed systems design will also benefit from it. The prerequisites are: intermediate level of knowledge/experience in microservice design, familiarity with domain modeling, and willingness to discuss software architecture.

III. Objective and Takeaways

Many have suggested using DDD to help define the functional scope of microservices. The main goal of this tutorial is to make clear how to apply this idea in practice. We will look at patterns, technologies, and good practices to create microservice architectures with better availability, scalability, reliability, and modifiability. Teaching DDD is not a goal, but we will cover basic DDD concepts.

The key takeaway messages of this tutorial are:

- DDD can help with defining microservices.
- The key DDD concepts for microservice design are: aggregate, bounded context (BC), domain event, and anticorruption layer.
- A service (e.g., a REST service) can have the scope of a microservice.
- A microservice can have the scope of a BC.

978-1-7281-4659-1/20/$31.00 ©2020 IEEE
DOI 10.1109/ICSA-C50368.2020.00010

- For inter-microservice (i.e., inter-BC) interaction, we can use domain events with asynchronous messaging, API calls, and service data replication.

IV. Tutorial Details and Implementation

A. Duration, structure, and schedule

This tutorial is designed as a full-day interactive session. It is organized with the following sections:

- **Brisk introduction to DDD**: the main concepts specifically relevant for microservice design: domain and domain model, entity and value object, aggregate, bounded context, context map and anticorruption layer, ubiquitous language, application service, domain service, repository, and domain event.
- **DDD modeling exercise**: hands-on exercise where participants work in groups of 4-6 (or in pairs depending on the classroom layout). They are given a problem and are asked to discuss and create a domain model using DDD concepts. Presenters oversee the exercise by playing the roles of domain expert and DDD specialist. At the end, domain models created by the participants are discussed in class.
- **Size of a microservice**: how granularity in terms of functional scope can affect the runtime behavior of microservices; microservice definition and design constraint discussion; practical examples.
- **DDD and microservice design**: looking at a possible solution for the DDD modeling exercise, we propose a microservice architecture covering five different scenarios. The first scenario is for a BC with a single aggregate. The second scenario shows a BC with multiple aggregates that need to interact. The third scenario shows multiple BCs that need to interact, and they use domain events for doing so. The fourth scenario has also different BCs, but they interact via API calls and an anticorruption layer. The last scenario shows different BCs but we use data replication with anticorruption layer in the form of an ETL component. For each scenario, we see the microservice design down to types of components and connectors using specific implementation technologies as examples.
- **Final exercise and wrap-up**: we finish with a fun Kahoot! exercise and discussion of the key takeaways.

B. Presentation approach and requirements

This tutorial is primarily a slide-based presentation. The topics covered are illustrated with examples. Questions and comments are stimulated at all times. The modeling exercise is a hands-on design exercise where participants put in practice the concepts just covered in the slides.

V. Presenters' Background

Paulo Merson has been programming in the small and programming in the large for over 30 years. He is a software developer at the Brazilian Federal Court of Accounts. He is also a Visiting Scientist with the Software Engineering Institute (SEI), a certified instructor for Arcitura, and a faculty member of the master program in Applied Computing at University of Brasilia. Paulo often delivers professional training to fellow developers in the United States, Latin America, and Europe. His speaking experience also includes tutorials at JavaOne, SPLASH/OOPSLA, SD Best Practices, SATURN, Dr. Dobb's Architecture & Design World, The SOA and Cloud Symposium, lectures to graduate students in different universities, and invited talks at different companies. He is co-author of Documenting Software Architectures: Views and Beyond, 2nd edition. Paulo holds a Bachelor of Science in Computer Science from University of Brasilia and a Master of Software Engineering from Carnegie Mellon University.

Joe Yoder (agilist, computer scientist, speaker, and pattern author) is the founder and principal of The Refactory, a company focused on software architecture, design, implementation, consulting, and mentoring on all facets of software development. Joe is also the president of The Hillside Group, a non-profit dedicated to improving the quality of life of everyone who uses, builds, and encounters software systems. Joe has presented many tutorials and talks, arranged workshops, given keynotes, and help organized leading international agile and technical conferences. He is best known as an author of the Big Ball of Mud pattern, which illuminates many fallacies in software architecture. Joe teaches and mentors developers on Agile and lean practices, architecture, building flexible systems, clean design, patterns, refactoring, software quality, and testing. Recently Joe has been working with organizations and thought leaders on the best practices for including quality aspects throughout the complete software life-cycle. Joe thinks software is still too hard to change and wants to do something about this. He believes using good practices (patterns), putting the ability to change software into the hands of the people with the knowledge to change it, and bringing the business side closer to the development process helps solve this problem.

Paulo and Joe created this tutorial over a year ago and have been evolving it ever since. A short version of the tutorial was presented at SATURN 2019 [12] and received very positive evaluations. A full-day version of the tutorial has been presented eight times to different microservice developers from various organizations, with spirited discussions in class and very positive feedback every time.

References

[1] E. Evans, *Domain- Driven Design.* Addison-Wesley, August 2003.

[2] V. Vernon, *Implementing Domain-Driven Design.* Addison-Wesley, February 2013.

[3] S. Millett and N. Tune, *Patterns, Principles, and Practices of Domain-Driven Design*. Wrox, April 2015.

[4] S. Wlaschin, *Domain Modeling Made Functional*. Pragmatic Bookshelf, January 2018.

[5] J. Lewis and M. Fowler, "Microservices." 2014. Available at martinfowler.com/articles/microservices.html.

[6] P. Merson, "Defining Microservices." SATURN blog, 2015. Available at insights.sei.cmu.edu/saturn/2015/11/defining-microservices.html.

[7] S. Newman, *Building Microservices*. O'Reilly Media, 2015.

[8] E. Evans, "DDD & Microservices – At last, some boundaries!". Presentation at QCon London 2016. Available at infoq.com/presentations/ddd-microservices-2016.

[9] E. Evans, "Language in Context." Presentation at DDD Europe 2019. See infoq.com/news/2019/06/bounded-context-eric-evans.

[10] C. Richardson, *Microservices Patterns*. Manning, 2018.

[11] N. Ford, R. Parsons, and P. Kua, *Building Evolutionary Architectures*. O'Reilly, September 2017.

[12] P. Merson and J. Yoder, "Modeling Microservices with DDD". Presentation at SATURN 2019. Available at saturn2019.sched.com/event/LY5d/modeling-microservices-with-ddd. May 2019.