Emerging Trends Research

Domain-Driven Design

Author: Denny Cox
Class: S6-RB10
Semester: 6

Table of contents

Research Plan

Problem/opportunity

Since it is my first time making enterprise software it interested me to find out if Domain-Driven Design is an approach for software development that could be used in every type of project, since it focusses on the designs of the models in a domain. This could be very helpful to have a strategy you can always use from the start of a software project and could potentially have benefits like saving time or making the scope of the project easier to understand and have a better basis for a project. The disadvantages are also something I want to research. I am interested in how exactly this strategy is carried out in a project. Next to this also want to know if there are alternatives to Domain-Driven Design. In the end, I want to know if I can make good use of it myself with future software projects and try it out on my individual school project.

Research questions

The main question would be: How can Domain-Driven Design improve the development of an enterprise software project with a microservice architecture? Corresponding sub-questions are:

- How should Domain-Driven Design be applied?
- What advantages and disadvantages does using Domain-Driven Design have?
- What are alternatives to Domain-Driven Design?

Research methods

The strategies that I will be using are the library  , the field  and the workshop  . The research methods I want to apply for this research are literature study, best, good & bad practices, prototyping and problem analysis.

Deliverables

For the deliverables of the research, I will deliver a research report where I work out the theoretical outcomes of the questions. Next to that I will also implement the outcomes to validate the findings with implementing it myself in my individual project. This would be prototypes in the form of diagrams or drawings.

Time estimation

The research for the emerging trends is done next to my individual project so I will not be working on it full time. In total I expect to work 1 sprint (3 weeks) on the research.

Research

How should Domain-Driven Design be applied?

Literature study

It's important to have a clear understanding of the term Domain-Driven Design (DDD) before researching how it should be applied. First the search plan for the question. The keywords that will be used are: Domain-Driven Design, DDD, enterprise software, software and development. The definition of DDD is: designing software based on models of the underlying domain (Fowler, 2014). It helps with the communication between the software developers and the domain experts. Bounded Context is also an important term when talking about DDD. In DDD the complete system gets divided up in Bounded Contexts like shown in the figure [1] below. As shown the bounded contexts themselves are separate with their own unrelated concepts like the ticket, but also have shared concepts like customer and product.

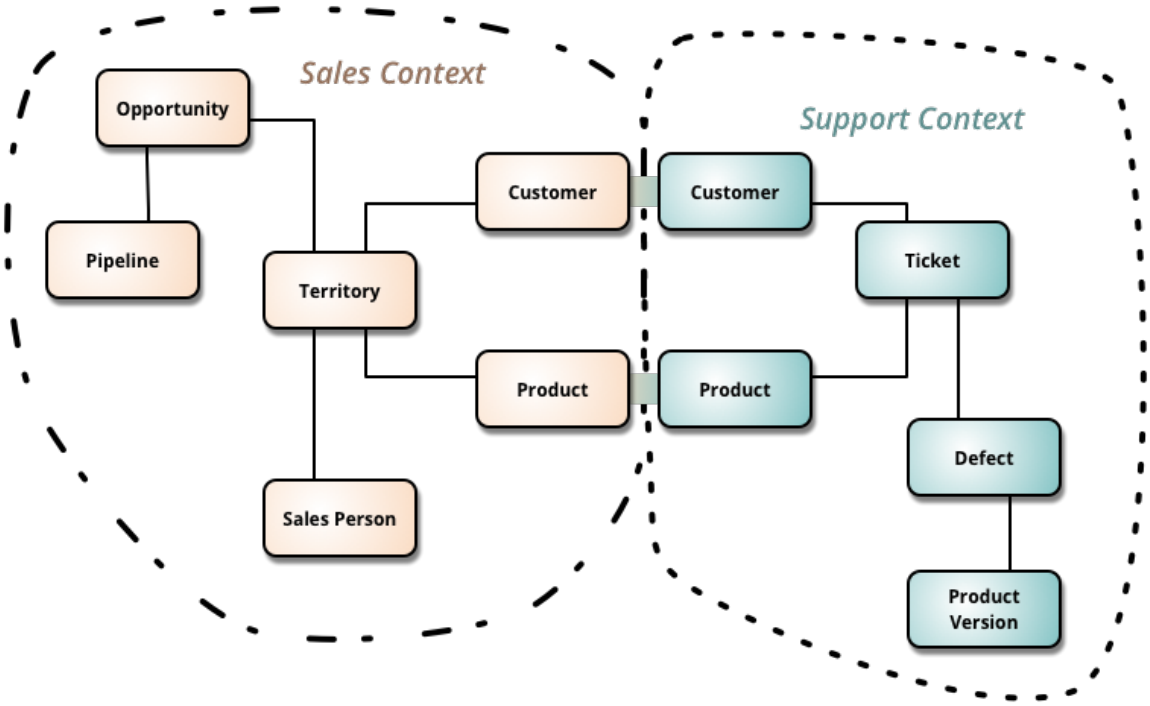

Figure 1 Bounded Contexts

Problem analysis

For the enterprise software Kwetter project that will be built there has to be a division of models that will be used for the implementation. The problem analysis requires asking questions. **Who** will apply DDD? The answer to this the developers and the domain experts. **What** will DDD add to a software system? The answer is clear contexts of the entire system. **Why** should DDD be applied? The answer to this question is that with enterprise software in particular a lot of bounded contexts exist, and they all must be understood to be implemented correctly. **When** should DDD be applied? The answer to this question is right from the start of a project. **Where** should DDD be applied? The answer to this question is as soon as a business becomes complex, one of the big challenges become to ensure the code does what the business wants to It is perfect for applications that have complex business logic. And this is

the sweet spot of DDD. **How** should DDD be applied? The answer to this question is with creating bounded contexts of the system.
As described, there are already existing solutions to apply DDD in enterprise software, it is not a new problem. This will be worked out further in this research document.

Best, good & bad practices

Best practices of applying DDD: always start with the requirements and modeling, by asking lots of questions. This way the scope of the project is clear right from the start and will not result in unexpected changes later on in development (Kavtaskin, 2016).
Good practices of applying DDD: good practices were mentioned in the literature study for a good implementation of DDD. Added to this, it is a good practice to set clear boundaries of a specific context in the system so that it has a single responsibility. This can save a lot of discussion time in the future (Croës, 2018).
Bad practices of applying DDD: not including domain experts, doing DDD for the sake of DDD, limiting to internal contexts only, not the entire ecosystem. These practices can lead to not fully understanding the scope of the entire system as it will be or implementing DDD when not really needed like in small projects (Alokmishra, 2022).

Prototyping

Implementing DDD in the individual Kwetter project different domains were decided on with reverse brainstorming. With this all the functionality of the system was written down on post-it notes and divided between domains. These are Authentication, Profile, Tweet and Moderation. The outcome of the reverse brainstorming can be seen in the figure [2] below.

Figure 2 Reverse brainstorming

After deciding on the domains (bounded contexts) themselves the next step was to work out what the concepts of each of them were. This will result in a class diagram that can be implemented during development. Authentication eventually became an account class. This can be seen in the figure [3] below. The most important domains of Kwetter now are account, profile, and tweet.

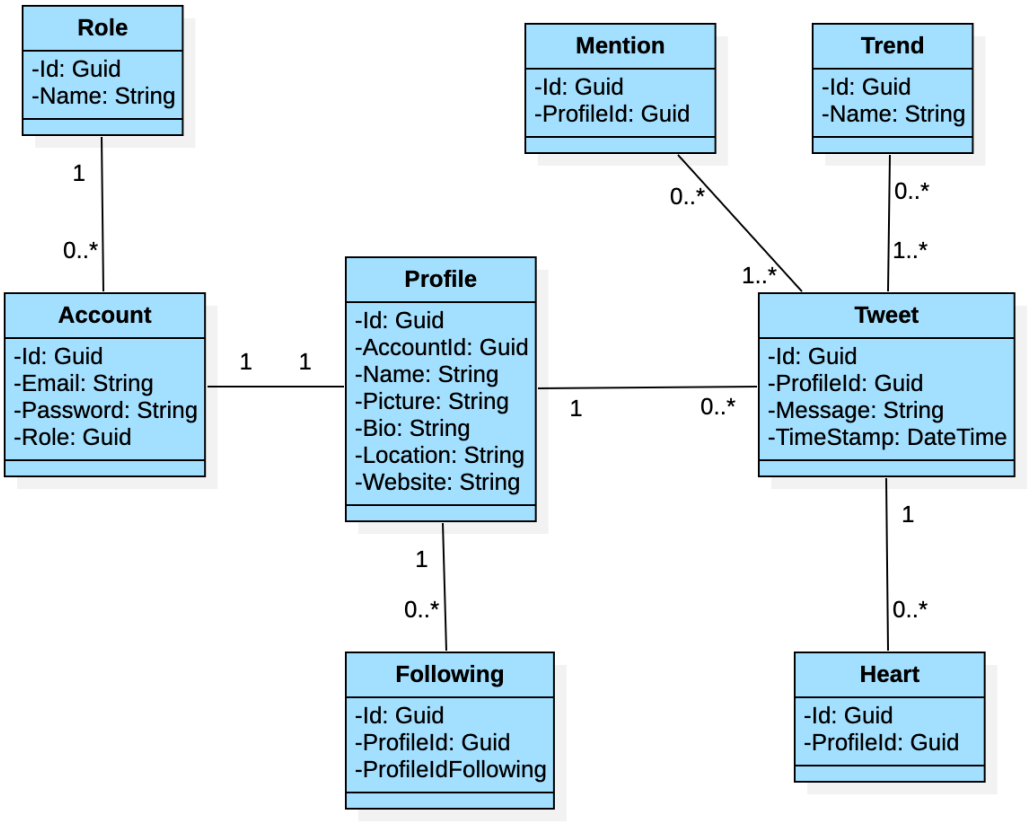

Figure 3 Class diagram

After this, it was decided that the account and role functionality would be combined in a Authentication microservice, the profile and following functionality would be combined in a Profile microservice, the tweet with heart, mention and trend functionality would be combined in a tweet microservice. The moderation would make use of the role functionality in all the different services to have all functionality available as a moderator. It is now ready to be implemented in software.

What advantages and disadvantages does using Domain-Driven Design have?

 **Literature study**

The biggest advantage of DDD is to create well defined components with clear contracts between them. This helps divide work and responsibilities and makes replacing and updating one of these components much easier with less impact on the overall system.

The key disadvantage is this assumes the people who are implementing it have a clear idea of the sort of solution a software developer/team are trying to produce. When there is a team of experienced developers who have written these sorts of systems many times before, this forward planning work well. However, if there is a relatively new system, with requirements very different to those a team is used to, attempting to create the design could be hard.
In general, DDD is sometimes also viewed as too complex to implement. If something is a bounded context or not could also be debatable and thus take a lot of time to get right and get everyone from the software team on the same page.

What are alternatives to Domain-Driven Design?

 **Literature study**

DDD doesn't really have an alternative for enterprise software development, the alternative rather is not using DDD if it is overkill for the size of the project at hand. But this would only be the case for monolithic applications.

Bibliography

Alokmishra, V. (2022, March 4). *DDD Anti-patterns: 5 things we get wrong with Domain Driven Design in practice*. Retrieved from Alok Mishra: https://alok-mishra.com/2021/11/03/ddd-anti-patterns/

Croës, G. (2018, June 20). *DDD 101 — The 5-Minute Tour - The Coding Matrix*. Retrieved from Medium: https://medium.com/the-coding-matrix/ddd-101-the-5-minute-tour-7a3037cf53b8

Fowler, M. (2014, January 15). *BoundedContext*. Retrieved from martinfowler.com: https://martinfowler.com/bliki/BoundedContext.html

Kavtaskin, Z. (2016, September 24). *Domain-Driven Design, My Top 5 Best Practices*. Retrieved from CodeProject: https://www.codeproject.com/Articles/1131462/Domain-Driven-Design-My-Top-Best-Practices