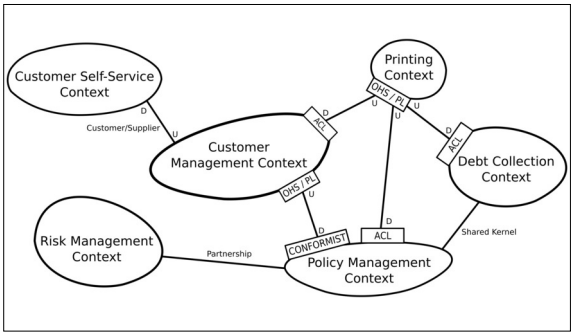
Stefan Kapferer

Student	Stefan Kapferer
Examiner	Prof. Dr. Olaf Zimmermann
Subject Area	Software and Systems

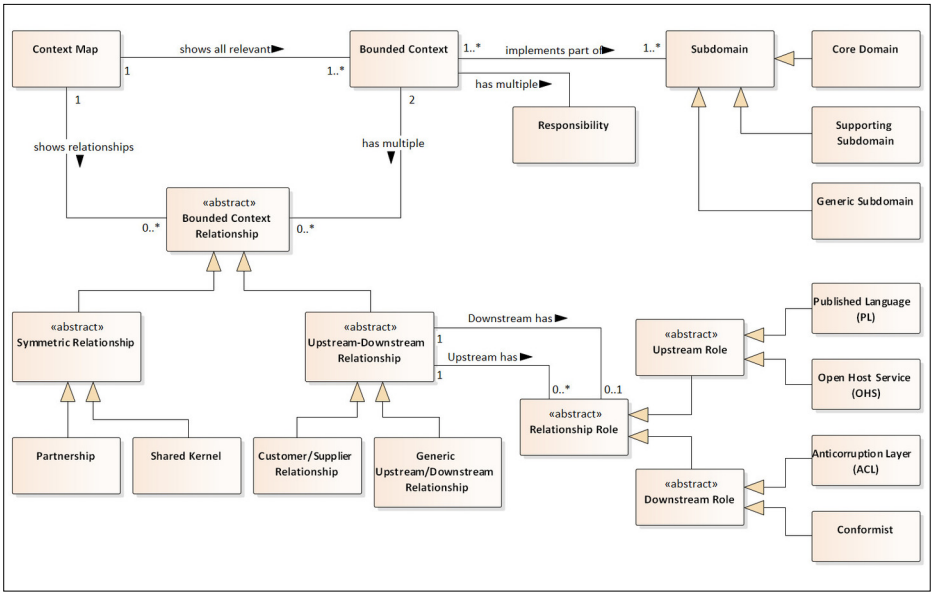
A Domain-specific Language for Service Decomposition



Example Context Map for a fictitious Insurance Scenario (Graphical Representation)

```
ContextMap {
    CustomerSelfServiceContext -> CustomerManagementContext : Customer-Supplier

    CustomerManagementContext -> PrintingContext : Upstream-Downstream {
        upstream implements OPEN_HOST_SERVICE, PUBLISHED_LANGUAGE
        downstream implements ANTICORRUPTION_LAYER
    }

    PrintingContext <- PolicyManagementContext : Upstream-Downstream {
        upstream implements OPEN_HOST_SERVICE, PUBLISHED_LANGUAGE
        downstream implements ANTICORRUPTION_LAYER
    }

    RiskManagementContext <-> PolicyManagementContext : Partnership

    PolicyManagementContext -> CustomerManagementContext : Upstream-Downstream {
        upstream implements OPEN_HOST_SERVICE, PUBLISHED_LANGUAGE
        downstream implements CONFORMIST
    }

    DebtCollection -> PrintingContext : Upstream-Downstream {
        upstream implements OPEN_HOST_SERVICE, PUBLISHED_LANGUAGE
        downstream implements ANTICORRUPTION_LAYER
    }

    PolicyManagementContext <-> DebtCollection : Shared-Kernel
}
```

Extract of the Insurance Example Context Map written in the developed Domain-specific Language (DSL)



Language Domain Model based on Strategic Domain-driven Design (DDD) Patterns

Introduction: Microservices have gained huge attention in the industry and in academia over the last years. Companies are adopting microservice architectures in order to increase agility, maintainability and scalability of their software. At the same time, decomposing an application into appropriately sized services is challenging. With its strategic patterns and especially Bounded Contexts, Domain-driven Design (DDD) provides an approach for decomposing a domain into multiple independently deployable services. However, existing modeling tools supporting DDD mainly focus on tactical DDD patterns. Not many approaches to a formal definition of the strategic patterns exist, and there are different interpretations and opinions regarding their applicability.

Result: This project presents a Domain-specific Language (DSL) based on the strategic DDD patterns. The model behind the language and its semantic rules aim to provide one concise interpretation of the patterns and how they can be combined. The DSL concept offers a tool to model a system in an expressive way, using the DDD language. With the implemented Service Cutter integration, we further provide a proof of concept showing how the DSL can be used as input for structured service decomposition approaches. The presented results and our evaluation of this approach illustrate the capabilities of DDD-based models for service decomposition. The developed tool offers an additional transformation to create PlantUML diagrams as an example of a graphical representation.

Objective: The DSL is meant to provide a foundation for other service decomposition approaches. Future projects may propose architectural refactorings for the DSL based on model transformations. Other approaches based on algorithms and heuristics similar to Service Cutter could be applied as well. A code generator to create microservice project templates for the modeled Bounded Contexts might be another promising future feature.