Bringing Context Mapper to the Developer's Workflow

Enhanced IDE Integration and Tooling Support

Graduate

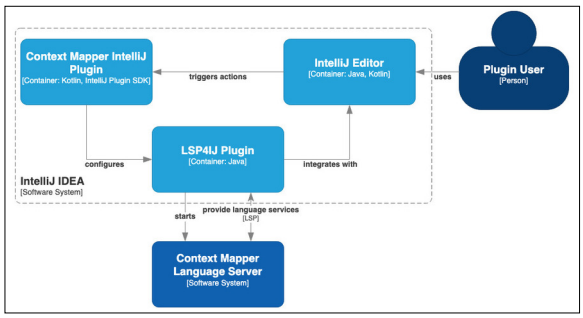
Lukas Streckeisen

Introduction: Context Mapper provides a Domain-Specific Language (DSL) for modelling software systems using Domain-Driven Design (DDD) patterns. The Context Mapper DSL (CML) language supports patterns from strategic and tactic DDD, as well as Value-Driven Analysis and Design. The current implementation of Context Mapper is based on Xtext, a Java-based framework for creating DSLs. Using Xtext, Context Mapper offers plugins for Eclipse and VSCode.
However, IntelliJ has become a popular development environment among Java developers, for which Context Mapper does not yet offer editor support. This situation is inconvenient for developers using IntelliJ and potentially prevents the widespread adoption of Context Mapper.

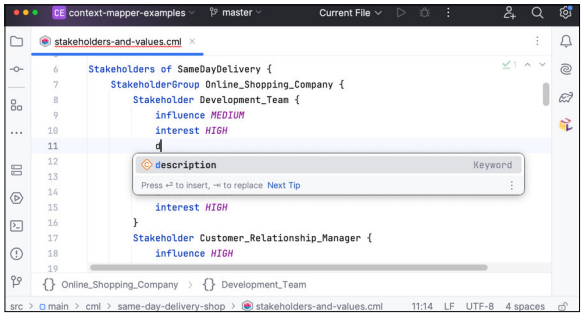
Objective: The goal of this thesis is to enhance the workflow for IntelliJ users, by developing a proof of concept (PoC) for a Context Mapper IntelliJ plugin. To achieve this goal, this thesis provides an overview of current language workbenches (frameworks for creating DSLs) and options for integrating DSLs in IntelliJ. From these technologies, the thesis evaluates the most suited technology to develop the PoC. The thesis further outlines how Context Mapper features beyond the project scope can be implemented and migrated in the future.

Result: The implemented plugin uses LSP4IJ, an open-source IntelliJ plugin based on the Language Server Protocol (LSP), and Langium, a TypeScript DSL framework. With LSP4IJ in charge of interactions between the language server and IntelliJ, the Context Mapper plugin itself simply configures LSP4IJ. All feature logic is implemented in the language server, which gives Context Mapper the flexibility to target other development environments in the future. The proof of concept successfully implemented important editor features, such as syntax highlighting, hyperlinking, autocomplete and a PlantUML component diagram generator. Future work includes providing a Java library for reading and writing CML models in an automated manner and therefore migrate the Context Mapper discovery library, CLI and ArchUnit extension.

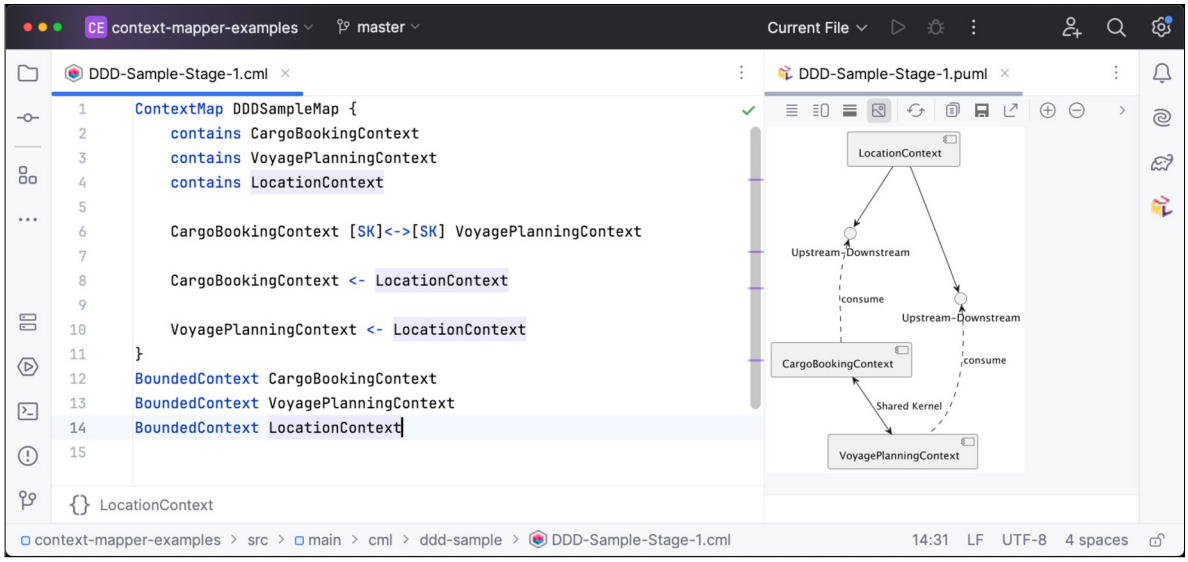
C4 container diagram of the implemented PoC
Own presentment



Autocomplete in the IntelliJ editor for the CML language
Own presentment



IntelliJ editor with a generated PlantUML component diagram
Own presentment



Advisor
Stefan Kapferer

Co-Examiner
Roman Blum, Swisscom Schweiz AG, Zürich, ZH

Subject Area
Software Engineering

Project Partner
IFS - Institute for Software, Rapperswil, SG