Modeling and Auto-Layouting of Ontological Software Architecture Views

Institute of Software Engineering
Software Quality and Architecture Group
Sandro Speth, M.Sc.

Description of a Practical Course (Fachpraktikum) in M.Sc. Inf and M.Sc. SE

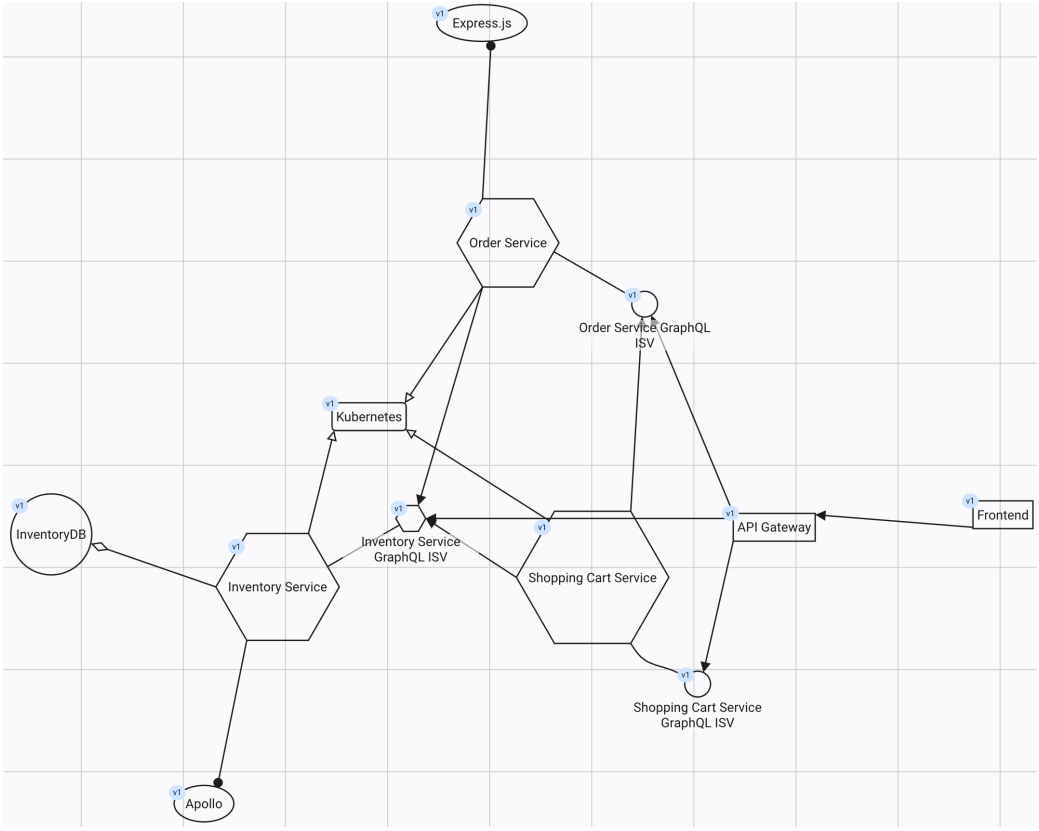

Figure 1: Excerpt of a web shop's architecture in Gropius.

1. Course Background

The architecture of a software system can be modeled linguistically or ontologically. While both approaches have advantages and disadvantages, ontological software architecture descriptions provide more flexibility in representing software systems' structural and semantic blueprints as mechanisms like templates define the types of software components and their relations. Examples of ontological software architecture models are TOSCA, the Palladio Component Model (PCM), or the metamodel of the Gropius issue management system.

One significant challenge in working with ontological software architectures is layouting the models as semantics might be defined at modeling time of the concrete architecture in contrast to the fixed semantics of a linguistical model which are defined when creating the metamodel. These graphs can become exceedingly complex with numerous nodes and edges denoting different types of components and their relations. While manual layouting works well for small architectures, it is impractical for larger ones, leading to the necessity for auto-layouting techniques. Auto-layouting algorithms aim to automatically organize the graphical representation in a way that is both aesthetically pleasing and functionally informative, ensuring readability and manageability of the architecture. However, auto-layouting becomes challenging for more complex ontological models. Some resulting graphs are planar, but others are not, especially when multiple component types and relations come into play. Some architectures become denser, e.g., if many components communicate with a pubsub component. Identifying and reacting to different architectural characteristics is difficult for existing auto-layout algorithms.

2. Tasks

This Practical Course focuses on the modeling and esp. auto-layout of ontological metamodels. As the primary use case, we focus on the Gropius metamodel and software architectures modeled with it. The graphical syntax of Gropius models looks similar to a UML component diagram, i.e., components might have provided and required interfaces and are connected with arrows. The shapes (and therefore also semantics) depend on the concrete template. Figure 1 depicts an example model of an excerpt of a microservice architecture that includes libraries and an infrastructure component.

The Practical Course consists of multiple tasks and milestones. The tasks briefly look as follows:

- Familiarize yourself with the Gropius metamodel.
- Create different architectures in the Gropius metamodel.
- Research on different layout algorithms and discuss their advantages and challenges.
- Develop one or multiple concepts to layout software architectures modeled in Gropius. Thereby, existing layout algorithms can be combined, or different algorithms can be developed to react to various cases.
- Implement one or multiple of your algorithms directly in a fork of the Gropius frontend or with a drawing library of your choice.
- Evaluate your algorithms with multiple given example software architectures, and optionally, your own examples.
- Document and explain your algorithm either as markdown wiki or in LaTeX

3. Examination

The grade consists of two parts: (1) a final presentation of each participant, (2) the solution's code, code quality, and documentation. All source code must be developed under an open-source license, e.g., MIT license. The students take the course and are graded individually.

4. Precondition

There are no mandatory preconditions regarding courses that must be taken upfront. Nevertheless, a basic understanding and knowledge of software architecture and modeling are recommended. Especially, the “Software Architecture” course will be helpful but not required.

5. Language, ECTS, Examiner, and Supervisor

This Practical Course belongs to the “Practical Course in Software Quality and Architecture” module (“Fachpraktikum Softwarequalität und -architektur”) which covers various topics around software architecture and its quality. It has a value of 6 ECTS points, i.e., 180 hours.

This Practical Course is conducted in ***German*** and is open for students in the master courses “Informatik” and “Software Engineering” (and its predecessor “Softwaretechnik”).

Examiner: Prof. Dr.-Ing. Steffen Becker; steffen.becker@iste.uni-stuttgart.de
Supervisor and Lecturer: Sandro Speth, M. Sc.; sandro.speth@iste.uni-stuttgart.de