[Extended Abstract] SQuAT-Vis: Visualization and Interaction in Software Architecture Optimization

Sebastian Frank
sebastian.frank@iste.uni-stuttgart.de
University of Stuttgart, Germany

André van Hoorn
van.hoorn@informatik.uni-stuttgart.de
University of Stuttgart, Germany

1 Motivation

Optimization of software architectures is a complex task that can not be fully automated. For this reason, software architecture optimization approaches, like SQuAT [4], often require human architects to participate in the optimization process, e.g., by selecting architectural candidates.

2 Related Work

Nevertheless, most of these approaches fail to support architects in solving their tasks as they provide no or insufficient visualization and interaction techniques. Regarding architectural information, architects can use graphical modeling editors, but they are not designed to compare multiple architectures. Regarding goal satisfaction, architects can use generic visualization tools for multivariate data, like the Trade Space Visualizer [1]. However, this requires architects to have visualization skills and invest time and effort, making interactive software architecture practically infeasible.

3 Idea

For this reason, we developed SQuAT-Vis - a tool that can visualize both architectural and goal satisfaction data and is equipped with interaction techniques. Despite our focus on compatibility with the technologies of SQuAT [4], SQuAT-Vis is conceptually compatible with other software architecture optimization approaches. The (current) limitations are rather technical ones, as only instances of the Palladio Component Model (PCM) [2] can be imported and the communication is Java-based.

4 Use Cases

We designed SQuAT-Vis with four typical use cases for software architects in mind. Firstly, selecting a single candidate or a subset of candidates from a population of architectural candidates. Secondly, the decision of whether to accept the proposed (intermediate) solutions or to continue. Thirdly, identify which architectural changes are necessary to implement a particular candidate. Finally, exploring and explaining the solutions to gain new insights and approve the proposed changes' usefulness. It has to be noted that these use cases are sometimes part of the optimization strategy itself, e.g., evolutionary optimization is iterative by definition and requires stopping criteria, but in other cases, they are implicitly left to the architect, e.g., she could restart a non-iterative process by providing the previous results as inputs. We based our selection of use cases on the general optimization process in the domain described by Aleti et al. [3], which consists of three recurrent phases: generation of new architectural candidates, evaluation of the candidates, and checking the satisfaction of the stopping criteria.

5 Visualization & Interaction

SQuAT-Vis follows the principle of connected views to achieve its goals. Despite administrative views, three views consisting of multiple visualizations are available. The Population View utilizes a scatter plot matrix to give an overview of the whole population. The Candidates View is designed for the comparison of a small number of candidates using radar charts. The Architecture View uses node-link diagrams and parallel coordinate plots to display architectural information regarding component usage, component dependencies, allocation, and resources. Candidates are automatically tagged and visually highlighted if they are Pareto-optimal, suggested by the software architecture optimization approach, or initial candidates. A toolbar allows the architect to group, control, and highlight candidates throughout all views.

6 Prototype

The front-end of SQuAT-Vis uses the JavaScript library D3.js [8] to display the data in the user's browser. The back-end is implemented using the Java Enterprise Edition. This design allows the architect to run SQuAT-Vis either as a local or remote server. The communication is based on the Java Sockets API, and a Java library is provided to support the implementation of client-side communication. Thus, assuring a loose coupling between SQuAT-Vis and the client-side software architecture optimization approach.

7 Evaluation

We conducted an expert user study to evaluate the usefulness of SQuAT-Vis qualitatively. Two experts and one non-expert in the field had to solve four tasks based on the previously described use cases. We provided two datasets based on the Extended Simple Tactics (ST+), and the Common Component Modeling Example (CoCoME). The results indicate that SQuAT-Vis provides sufficient support for selecting candidates and deciding on the optimization process's termination. Therefore, we see SQuAT-Vis as an important contribution to (interactive) software architecture optimization.

8 Pointers & Talk

SQuAT-Vis has been developed as part of a Master's Thesis [5]. A tool demonstration paper [6] (including a video [7]) will be presented at ECSA 2020. In this talk, we will give an overview of the mentioned concepts, methodology, and evaluation, as well as a short demonstration of the tool.

9 References

[1] G. Stump et al. "The ARL trade space visualizer: An engineering decision-making tool". In: MA&O. AIAA, 2004, p. 4568

[2] S. Becker, H. Koziolek, and R. Reussner. "The Palladio component model for model-driven performance prediction". In: JSS 82.1 (2009), pp. 3–22

[3] A. Aleti et al. "Software architecture optimization methods: A systematic literature review". In: TSE 39.5 (2013), pp. 658–683.

[4] A. Rago et al. "Distributed quality-attribute optimization of software architectures". In: SBCARS. ACM. 2017, p. 7.

[5] S. Frank. "Techniques for Visualization and Interaction in Software Architecture Optimization". MA Thesis. University of Stuttgart, 2019.

[6] S. Frank and A. van Hoorn. "SQuAT-Vis: Visualization and Interaction in Software Architecture Optimization". In: ECSA. 2020.

[7] Frank, S.: SQuAT-Vis Showcase Video, https://youtu.be/YUGujyR0jA8

[8] M. Bostock.D3.js.