DOER

Description

Project title: Software architecture viewpoint design and visualisation tool

Description
In software engineering, the process of transforming textual architecture documentations into some diagrammatic or visualisation representations is inevitable for architects and project managers, this could lead to numerous components and models which emphasise distinct focal points regarding different stakeholders and contexts involved in the development (i.e. different architecture viewpoints). In the industrial environment, the static architecture diagram (drawn on the whiteboard or using generic diagram design tools) may be transient and lack visualisation or software engineering support, while professional modelling and analysing tools may be difficult to utilise and thus lead to a large amount of workload. On the other hand, existing software architecture design and visualisation tools (e.g. Structurizr, Confluence) can satisfy most of the above requirements, but are rarely supported with the features of multiple viewpoints switching and connections revealing.

This project aims to provide a composite software architecture viewpoint design and visualisation tool with customised elements designers:

- For a single architecture view, it will help users design a single architecture diagram/visualisation product with customised components and connectors.
- For multiple architecture views, it will help users reveal the conceptual and/or realistic connections between different components under different viewpoints.

In addition, this tool will provide a user-friendly interface and support simple operations (e.g. drag-drop and text input) on core functions with some extension tools used for different software engineering behaviours. The final software artefact of this project could be a web application that has a strong front-end that focuses on data representation and visualisation, and a light-weight back-end that focuses on data storage.

Objectives

Primary objectives: Software architecture = {Elements, Forms, Rationale/Constraints}

1. Implement an architecture component design tool that provides predefined components for users to build customised architecture components. (progress: single-level architecture component)
2. Support hierarchical features on architecture components. (progress: hierarchical architecture component)
3. Provide visualisation on architecture components.

4. Implement an architecture connector design tool that provides interactions among architectural components. (revealing the relationship by component definitions and

supporting manually connection) (progress: hierarchical architecture component with intra-components connector)
5. Support hierarchical features on connectors. (progress: a single architecture view - hierarchical architecture component with inter-components connector)
6. Provide visualisation on connectors.

7. Implement a design tool that is able to edit and connect multiple views. (progress: composite views with view-view connections)
8. Support intra-components and inter-components connection. (progress: customised model: composite views with hierarchical view connections)
9. Based on connections, the tool is able to reveal constraints between different components and views. (progress: core function)

10. Conduct a user evaluation on the core tool, evaluate the design functions and the visualisation design.

Secondary objectives: improve user experience & software quality

11. Improve user experience based on the first evaluation
12. Add IO function: the product diagram / visualisation can be imported or exported for reuse.
13. Implement extension tool(s) that provide further software engineering behaviours. (e.g. Agile developments, existing software engineering tools connection)
14. Conduct the second user evaluation on the overall tool, evaluate the quality and robustness of the system.

Tertiary objectives: support further user behaviours

15. Improve software quality furthermore based on the second evaluation.
16. Support collaborations on the tool, enable version control or multi-user editing on a single architecture.
17. Provide API for users to design customised extension tools.

Ethics

Based on the objective 10 and 14, two separate user evaluations need to be conducted. These user evaluations do not need any personal information from users, but only collect their advice to further optimise the software. The artifact evaluation form is signed and submitted through MMS together.

Resources

This project does not require any special hardware, software and licenses. All designs and implementations requirements can be fulfilled by current lab machines.