Project Proposal

Software Architecture

Semester 1, 2022

Richard Thomas & Brae Webb

Project Context

Throughout the software architecture course, you have learnt about a subset of quality attributes of concern to software architects. You have also been exposed to a number of techniques to satisfy these attributes. Now, as the capstone project, you are required to

- propose a non-trivial software project,
- identify the primary quality attributes which would enable success of the project,
- design an architecture suitable for the aims of the project,
- deploy the architecture, utlising any techniques you have learnt in or out of the course, and
- evaluate and report on the success of the software project.

The successful completion of the project will result in three deliverables, namely,

i a proposal of a software project, the proposal must clearly indicate and prioritise two or three quality attributes most important to the project's success,

ii the developed software, as both source code, and a deployed artifact, and

iii a report which evaluates the success of the developed software relative to the chosen quality attributes.

Your software deliverable will include all supporting software (e.g. test suites or utilities) that are developed to support the delivered software.

1 Introduction

We have looked at several core quality attributes in this course, and will continue to look at more over the remainder of the semester. These attributes were selected because they are key concerns of many real-world software projects. In this project, we will have an opportunity to explore some of the fun of industry. You will take the role of an entrepreneur, software architect, developer, and operations team.

Your first role as an entrepreneur will be to use your creativity to think of a software project that interests you. Your proposed project does not have to be profitable, nor does it have to be unique. If you are struggling to think of a project, consider what annoys you in your day-to-day life. Consider if software might help ease the annoyance. Alternatively, look at existing everyday software like Netflix, TikTok, VS-Code, or others. You are welcome to create off-brand versions of any existing software, there are no marks for whether the software would be profitable or successful. The lone requirement of your project is that, to function appropriately, it must demonstrate two or three of the quality attributes explored in this course[1].

[1] No, simplicity is not allowed.

Briefly, these attributes are:

Availability The software can always be accessed by end users, either at any time or on any platform, or both.

Deployability The required computing infrastructure for the software can be easily provisioned, including updating both the infrastructure and the software.

Extensibility Features or extensions can be easily added to the software over its lifespan.

Interoperability The software can easily share information and exchange data with internal components and other systems.

Maintainability The software is designed to be cost effectively modified over its lifespan.

Modularity Components of the software are separated into discrete modules.

Reliability The software consistently delivers its functionality without failure. You would need to define what "consistently" means for your system and how it will be measured.

Scalability The software is simultaneously usable by a large number of end users and is economical to deliver with varying user loads.

Security Software that maintains normal operations and functionality even when subjected to attacks. Systems and resources in its environment remain safe and the attacks are detected and mitigated.

Testibility The software is designed so that automated tests can be easily deployed. This is beyond just automated unit testing.

While security may be an appropriate quality attribute to use as the focus of your project, all software systems must be developed to be "secure enough" for the context. Conquently, it is expected that all projects will consider security, even if it is not fundamental to the project's success.

Once you have settled on a project, write up a proposal for the project, as described in section 2. Before you get too far writing your proposal, please try and discuss the idea with teaching staff, this will help ensure you do not have to re-write it from scratch.

2 Content

Your proposal will answer the following questions:

- What is your project?
- Which quality attributes are most important and why?
- If trade offs are necessary, which attributes have higher priority?
- What are the basic features you plan to implement?
- How will you evaluate whether your project has the important quality attributes?

The proposal should not exceed two pages. The suggested proposal structure is as follows.

Title Name for your project, get creative.

Author Your name and student number.

Abstract An elevator pitch to sell the project. This should highlight the quality attributes crucial to the project's success.

Functionality Summary of the features delivered by the complete software product. This is what would be delivered if you built the entire system. Use this to sell why your project is fun or interesting.

Scope Description of the fundamental functionality to be delivered as the Minimum Viable Product (MVP)[2]. This is what you have to implement so be realistic!

Quality Attributes A more detailed description of the quality attributes and why they are crucial to the project. They should be measurable and/or testable.

Evaluation Description of how you will evaluate whether your project has achieved the desired attributes. This is one of the most important parts of the proposal. It must be clear how the evaluation will be done, and it must be feasible.

3 Submission

The following are *important* details about how your proposal must be submitted. Read the following carefully, misreading or misunderstanding the requirements does not except you from them.

- Your proposal is due by 16:00 (AEST) on April 11. Late submissions will not be marked.
- Your proposal **must** be written in *markdown*[3].
- Submission of the proposal component of the assignment is via a GitHub repository[4].
- You have been provisioned a directory in the GitHub repository[5], where you should place your markdown files and any assets (images, code snippets, etc) that are included by the markdown files.
- Voting on proposals of interest closes at 16:00 (AEST) on April 19. If you do not nominate a reasonable number of projects by voting on them, you may be allocated to any project.

Below is a possible structure of your directory. `proposal.md` may have relative references to the images and files in the `assets` directory.

```
s4435400/
    proposal.md
    assets/
        module-structure.png
        plugin-example.js
```

[2] https://www.agilealliance.org/glossary/mvp/

[3] https://www.markdownguide.org/

[4] It is important that you are continually keeping GitHub up to date with your progress. Keeping up to date will avoid the merge traffic jam near the due date.

[5] https://github.com/CSSE6400/project-proposal-2022

4 Marking Criteria

Criteria	Standard				
	Advanced (5)	**Proficient (4)**	**Developing (3)**	**Emerging (2)**	**No Evidence (0)**
Functionality 20%	Full system functionality clearly and concisely describes a complete system. MVP is very well defined, clearly minimal, and clearly feasible.	Full system functionality is well defined and appears to describe a complete system. MVP is fairly well defined, appears to be minimal, and seems feasible.	System functionality is fairly clear but appears to be missing one or two aspects of the system. MVP is generally clear but lacks some important aspects or contains too many features, but still may be feasible.	System functionality is not very clear or is missing a few aspects of the system. MVP lacks some important information, or is too small or large, or is not feasible.	System functionality is vague or contradictory, or it is missing several aspects of the system. MVP lacks important information, or is far too small or large, or is not feasible.
Quality Attributes 35%	All quality attributes are clearly important, their selection is well justified, and there are no other obviously more important attributes. They are clearly measurable or testable.	All quality attributes are important, their selection is fairly well justified, and there are no other obviously more important attributes. They seem to be measurable or testable.	All quality attributes seem important, their selection is mostly adequately justified, and any other potential important attributes are not too much more important. Most seem to be measurable or testable.	Some quality attributes are important, their selection is weakly justified, and there appear to be other more important attributes. Most are not described in a way to indicate how they can be measured or tested.	Quality attributes are not important, or their selection is poorly justified, or there are clearly more important attributes. Their descriptions make it difficult to see how they can be measured or tested.
Evaluation 35%	Evaluation plan is clearly described and is clearly feasible. It covers all functionality of the MVP and all quality attributes.	Evaluation plan is fairly clearly described and seems to be mostly feasible. It covers almost all functionality of the MVP and all quality attributes.	Evaluation plan is comprehensible and does not appear to be too difficult to implement. It covers most functionality of the MVP and most quality attributes.	Evaluation plan is not clear or does not appear to be feasible. It covers some functionality of the MVP and at least the most important quality attribute.	Evaluation plan is confusing or contradictory or is clearly not feasible. It covers little functionality of the MVP or, at best, less important quality attributes.
Documentation 10%	The document structure leads the reader to a clear understanding of the proposal. It is at an appropriate technical level. Grammar and prose enhance the clarity of the document.	The document is logically structured. It is at an appropriate technical level. Grammar and prose are appropriate for a professional document.	The document structure does not hinder comprehension. It is mostly at an appropriate technical level. Grammar and prose do not hinder comprehension.	The document is not logically structured. It is mostly at an appropriate technical level. Grammar and prose hinder comprehension a little.	The document is poorly structured, requiring the reader to reference other sections to understand the content. It is not at an appropriate technical level. Grammar and prose make comprehension difficult.