Project Proposal

Software Architecture

Semester 1, 2024

Richard Thomas & Brae Webb

Project Context

During the software architecture course, you will learn about a subset of quality attributes of concern to software architects. You will be exposed to a number of techniques to satisfy these attributes. In 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, utilising 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 artefact, and

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

Your software deliverable includes 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, you 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 is 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, VSCode, or others. You are welcome to create off-brand versions of any existing software. There are no marks for whether the software is unique, or 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, some of 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. Consequently, 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 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 delivered its important quality attributes?

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

Title Name for your project, get creative.

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

Author Your name and student number.

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 15:00 on March 28. Late submissions will be penalised by one grade per 24-hour period. See the course profile[3] for details. The maximum extension length is 7 days.
- Your proposal **must** be written in *markdown*[4].
- Submission of the proposal component of the assignment is via a GitHub repository[5].
- You have been provisioned a directory in the GitHub repository[6], where you should place your markdown file and any assets (images, code snippets, etc) that are included by the markdown file.
- Your directory contains a template markdown file named `proposal.md`. Do ***not*** change the formatting of the template. Insert your proposal content under the headings provided.
- Only what is in your directory in the main branch at the submission deadline will be marked and made available for voting.
- Please validate that your proposal renders sensibly on the proposal website: `https://csse6400.github.io/project-proposal-2024/`
- You can view example proposals at `https://csse6400.github.io/project-proposal-examples/`.
- Voting on proposals of interest closes at 15:00 on April 15. 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://course-profiles.uq.edu.au/student_section_loader/section_5/132140

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

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

[6] https://github.com/CSSE6400/project-proposal-2024

Marking Criteria

Criteria	Standard						
	Exceptional (7)	Advanced (6)	Proficient (5)	Functional (4)	Developing (3)	Little Evidence (2)	No Evidence (1)
Functionality 20%	Full system functionality clearly and concisely describes a complete and coherent system. MVP is very well defined, clearly minimal and feasible.	Full system functionality is well defined and describes a complete system. MVP is well defined, clearly minimal, and seems feasible.	Full system functionality is fairly well defined and describes a mostly complete system. MVP is fairly well defined, close to being 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 is not minimal; could be feasible with adjustment.	System functionality lacks some clarity but the general idea of the system is still fairly clear. MVP idea is generally clear but lacks some important aspects or is too large.	System functionality is not very clear or is missing a few aspects of the system. MVP lacks important information, 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, is far too small or large, or is clearly not feasible.
Quality Attributes 35%	All quality attributes are clearly important, well justified, and there are no other obviously more important attributes. They are clearly measurable or testable.	All quality attributes are clearly important, fairly well justified, and there are no other obviously more important attributes. They seem to be measurable or testable.	All quality attributes seem important, adequately justified, and other potential important attributes are not too much more important. Most seem to be measurable or testable.	All quality attributes seem important, most are adequately justified, and few other potential important attributes are more important. Most seem to be measurable or testable.	Some quality attributes are important, some are 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.	Some quality attributes are important, some are 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.	Most quality attributes are not important, are 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. Covering all MVP functionality and all quality attributes.	Evaluation plan is clearly described and seems to be feasible. Covering all MVP functionality and almost all quality attributes.	Evaluation plan is fairly clearly described and seems to be mostly feasible. Covering almost all MVP functionality and most quality attributes.	Evaluation plan is comprehensible and seems to be somewhat feasible. Covering most MVP functionality and most quality attributes.	Evaluation plan is unclear or does not appear to be feasible. Covering some MVP functionality and at least the most important quality attributes.	Evaluation plan is unclear and does not appear to be feasible. Covering some MVP functionality and some of the most important quality attributes.	Evaluation plan is confusing or contradictory or is clearly not feasible. Covering little MVP functionality or, at best, less important quality attributes.
Documentation 10%	Document structure leads the reader to a clear understanding of the proposal. Technical level of text is always appropriate. Grammar & prose enhance the clarity of the document.	Document is logically structured. Technical level of text is appropriate. Grammar & prose are professional in nature.	Document is fairly logically structured. Technical level of text is mostly appropriate. Grammar & prose are mostly professional in nature.	Document structure does not hinder comprehension. Technical level of text is mostly appropriate. Grammar & prose do not hinder comprehension.	Document is not logically structured. Technical level of text is at times appropriate. Grammar & prose hinder comprehension a little.	Document is poorly structured, requiring referencing other sections to understand it. Technical level of text is mostly inappropriate. Grammar & prose make comprehension difficult.	Document is very poorly structured, making it difficult to follow. Technical level of text is inappropriate. Grammar & prose make comprehension very difficult.