Capstone Project

Software Architecture

Semester 1, 2022

Brae Webb & Richard Thomas

Summary

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 fundamental 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 must 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, you will have an opportunity to explore some of the fun of industry. You will take the role of an entrepreneur, software architecture, developer, and operations team.

As entrepreneur, you have proposed a project idea. These proposals have been evaluated by your peers and the teaching team. You have been allocated to a project based on interest you have indicated by voting on proposals and on your previous coursework experience.

As a team you now need to perform the roles of software architect, developer, and operations team. You should design the basic structure of the initial software architecture, based on the scope, functionality, quality attributes, and evaluation plan from the proposal. The details of the architecture are expected to evolve as you start implementing parts of the system.

Part of the assessment will be how the architecture evolves in response to what you learn during development. You need to write Architectural Decision Records (ADRs) for each decision you make about the design of the architecture [1]. These are to be recorded in your GitHub repository so that the marker can see how your architecture evolved and the reasons for the decisions you made.

2 Software

You need to implement a software system that delivers a Minimal Viable Product (MVP)[1]. The MVP needs to implement a usable core of the system's functionality, which demonstrates that the architecture could deliver the full system functionality. The MVP also needs to allow the software architecture to be tested to determine if it can deliver the project's important quality attributes.

You may renegotiate the scope of the system during the project, if you determine that certain aspects of the original scope are not feasible within the project time constraints. The earlier you do this, the less it will impact on your final result. You will not explicitly lose marks for renegotiating scope, unless the revised scope limits your ability to adequately test important quality attributes. But, late changes to scope are likely to have a flow-on effect that could reduce the quality of your final deliverables. This means that you should attempt to implement some of the riskier parts of the project early.

3 Evaluation

You need to test the software system that you implement to demonstrate how well its architecture supports delivering system functionality and its quality attributes. This evaluation should be based on the proposal's evaluation plan, but should *not* be limited to only what is in that plan. You will be assessed on how well you test your system in terms of both functionality and quality attributes. Discovering issues with the system or its architecture during testing will not adversely affect your marks for the evaluation component of the assessment.

A section of your project report needs to summarise the test results and provide access to the full suite of tests. You should automate as much of the testing as possible. Any manual tests need to be documented so that they can be duplicated. The results of all manual tests need to be recorded in a test report. This may be a section of the project report, or it may be a separate document with a link to it from the project report. You need to include test code and test infrastructure in your project's repository.

4 Report

The report should

- describe the architecture you implemented for the MVP,
- describe trade-offs made in designing the architecture,
- evaluate how well the architecture will support delivering the complete system, and
- summarise testing results.

The description of the architecture and its evaluation should be in a similar format to what was expected for the documentation assignment. You need to provide enough detail through views, diagrams, and commentary to give the reader a complete understanding of the architecture's design. You should describe parts of the detailed design that demonstrate how the architecture supports delivering key quality attributes.

Your evaluation of the architecture should discuss how well the architecture is suited to delivering system functionality and quality attributes. You should make use of test results to support some of your claims. In discussing advantages and disadvantages of the architecture, you should describe the rationale behind trade-offs made in the architecture design. You should summarise and/or reference ADRs that relate to important decisions that affected your architecture.

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

5 Repository

Your team will be provisioned with a repository on GitHub. All development work and documentation are to be committed to the repository. All project artefacts are to be submitted via this repository.

All ADRs are to be stored in the `/model/adrs` directory of your repository. Model artefacts (e.g. PUML or Structurizr DSL files) should be stored in the `/model` directory.

The report must be stored in the `/report` directory of your repository.

Do not commit large binary files to the repository. (i.e. Do *not* commit Word documents or frequently changing PDF files to the repository.)

6 Academic Integrity

As this is a higher-level course, you are expected to be familiar with the importance of academic integrity in general, and the details of UQ's rules. If you need a reminder, review the Academic Integrity Modules[2]. Submissions will be checked to ensure that the work submitted is not plagiarised.

All code that you submit must be your own work, or must be appropriately cited. If you find ideas from online sources (e.g. Stack Overflow), you must cite and reference[3] these sources. Use the IEEE referencing style[4] for citations and references. Citations should be included in a comment at the location where the idea is used in your code. All references for citations must be included in a file called `refs.txt`. This file should be in the root directory of your repository.

You may use libraries to help implement your project. The library's license must allow you to use it in the context of your project. All libraries used in your project must be listed in a file called `libs.txt`. This file should be in the root directory of your repository.

Uncited or unreferenced material will be treated as not being your own work. Significant amounts of cited material from other sources will be considered to be of no academic merit.

7 Marking Criteria

20% Extent to which project's scope was delivered.

20% Suitability of architecture to deliver system goals.

20% Quality and thoroughness of testing.

20% Clarity, accuracy and completeness of architecture's description.

20% Insightfulness of architecture's evaluation.

References

[1] R. Thomas, "Architectural decision records," March 2022. https://csse6400.uqcloud.net/handouts/adr.pdf.

[2] https://web.library.uq.edu.au/library-services/it/learnuq-blackboard-help/academic-integrity-modules

[3] https://web.library.uq.edu.au/node/4221/2

[4] https://libraryguides.vu.edu.au/ieeereferencing/gettingstarted