Reflection Report on OAR

Hunter Ceranic

April 15, 2024

1 Changes in Response to Feedback

Feedback from the reviewers (Domain Expert, Secondary Reviewer, and the Course Instructor Dr. Smith) was integrated into the documentation. Namely there was a shift in the projects overall tone to becoming more of a "Educational" tool, and hopefully that is reflected in the final version of the project.

1.1 SRS

Using the SRS document I was able to create a kind of roadmap for where the project was headed, and for me at least, really develop the themes of the project (those being designing the project to be an educational tool, and using the project as an opportunity to teach myself a lot about software development). Most of the updates over the course of the project related to adding more references or information to the Theoretical Models (especially since my understanding of them was very weak in the initial revisions), and updating descriptions or assumptions to be more clear. The issues created by the reviewers and their corresponding commits that resolve them are found in each issues comments.

1.2 Design and Design Documentation

I think the basic component modules that needed to be part of the design were clear from the beginning, however understanding how those modules needed to be connected, and how to communicate what they did on a high-level were more difficult. Even in the final version there are clear differences between the design documentation and how the actual implementation is handled, which I left different intentionally (I think the way its laid out in the documentation is easier to understand in that context, but it doesn't make as much sense in the actual code). Most of the updates over the course of the project related to adding more detail to the Module Interface Specification, further modularization of specific parts like the GUI and updates for clarity and specificity. The issues created by the reviewers and their corresponding commits that resolve them are found in each issues comments.

1.3 VnV Plan and Report

The VnV Plan was probably the document where I was the most ambitious, but also the document that was the most promising regarding my own learning. As someone who has never really done code testing outside of manual trial-and-error, learning about and integrating linters, coverage tools, pytest and automated CI/CD in general was a daunting task, but I really saw it as a challenge to improve my software skills. Overall the VnV Plan and report were probably both the most frustrating and simultaneously the most satisfying parts of the project (once I got them to work). I still don't think I completely documented these tests correctly, as writing the tests kind of took a backseat in my learning to actually getting them to even run, but I'm proud of how far I've come. Most of the document updates over the course of the project related to adding more tests to the VnV Plan, trying to figure out the scope (I originally removed coverage because I didn't think I would have time, then added it back in) and of course adding more clarity. The issues created by the reviewers and their corresponding commits that resolve them are found in each issues comments.

2 Design Iteration

The core of the final design was similar to the initial version, however the way modules were linked to each other were updated during the implementation. Module names were changed to match the file names.

3 Design Decisions

The decision to create a GUI for the project was so that I could ensure the proper images were being classified. The assumptions constrained the problem enough to ensure the project was focused on the goal of making it "educational" rather then simply a high performance classification model.

4 Reflection on Project Management

The specific tools used in this project were CircleCI [**?**], Flake8 [**?**], coverage.py and pytest.

4.1 How Does Your Project Management Compare to Your Development Plan

Due to issues integrating the test suite into CircleCI some test were dropped from the project in interest of time, namely portions of the Non-Functional Requirements Tests.

4.2 What Went Well?

CircleCI was great for testing build portability and linting. Additionally I learned a lot about CI/CD, Markdown, LaTeX, Makefiles, Config files, and other software development tools during the course of this project, and essentially taught myself how to use a lot of these things from scratch (which is probably why as the project progress you may also be able to track my competence using these tools).

4.3 What Went Wrong?

Integrating CircleCI and pytest was a pain. There was a lot of issues when it came to understanding how to link all the files properly and have the tests be recognized by CircleCI. It took way too long to figure that out and as a result a lot of the testing suffered, since I simply did not have enough time to cover it all. As a result, I dropped doing the comparison portion of the Performance Non-Functional Requirement Tests, as the actual performance of the model was least important to my learning overall, and also to the purpose of the project.

4.4 What Would you Do Differently Next Time?

Next time I would probably use GitHub Actions. From what others have said it seems a bit simpler to integrate pytest with, and that would save me a lot of time, and give me the ability to improve the quality of the software and documentation I wrote and the testing performed. That being said in hindsight a lot of my issues came from my lack of familarity with *all* of the tools involved so GotHub Actions might not have actually had an impact on how the project went as a whole. In general though, I am very happy with the amount of different tools and concepts I learned about through the course of this project, and the progress I think I've made as a software developer as a whole.

Thanks for reading, I really appreciated this project even though maybe it isn't as sophisticated or thorough as I would've liked it to be it still was exactly the type of course I was wanting to take in grad school!