Reflection Report on MTOBridge

Team 15, Alpha Software Solutions
Badawy, Adham
Yazdinia, Pedram
Jandric, David
Vakili, Farzad
Vezina, Victor
Chiu, Darren

[Reflection is an important component of getting the full benefits from a learning experience. Besides the intrinsic benefits of reflection, this document will be used to help the TAs grade how well your team responded to feedback. In addition, several CEAB (Canadian Engineering Accreditation Board) Learning Outcomes (LOs) will be assessed based on your reflections. —TPLT]

1 Changes in Response to Feedback

[Summarize the changes made over the course of the project in response to feedback from TAs, the instructor, teammates, other teams, the project supervisor (if present), and from user testers. —TPLT]

[For those teams with an external supervisor, please highlight how the feedback from the supervisor shaped your project. In particular, you should highlight the supervisor's response to your Rev 0 demonstration to them. —TPLT]

1.1 SRS and Hazard Analysis

1.1.1 SRS

1.1.2 Hazard Analysis

The changes made to the hazard analysis in response to feedback were relatively small. These changes mainly related to clarity and completeness, and had little impact on the final implementation. The largest change that was made was adding a new hazard relating to the case of lost saved data. Here is a complete list of changes:

1. Swapped the position of HA-11 and HA-12 for better readability

2. Clarified certain vague failure effects (e.g., "program cannot function")

3. Modified incorrect failure conditions in HC-2 and HD-2

4. Moved SR-3 and SR-4 to phase 2 of implementation

5. Changed some of the recommended actions in HA-1 to better resolve the identified failures

6. Clarified some of the recommended actions in HA-1, HA-6, and HD-3

7. Added background information in section 1

8. Added a new hazard for lost save data.

1.2 Design and Design Documentation

1.3 VnV Plan and Report

2 Design Iteration (LO11)

[Explain how you arrived at your final design and implementation. How did the design evolve from the first version to the final version? —TPLT]

Adham:

Darren:

David:

Farzad:

Pedram: Arriving at our final design was a process of rapid requirements gathering, feasibility analysis and software design built upon the initial findings and the proof of concept. In terms of requirements gathering, the client was regularly given multiple mockups of the proposed designs to choose from in addition to refining the requirements, constrains and assumptions. The team would often conduct brief feasibility analysis to better understand the priority and complexity of each ask. Very importantly, relational tools and diagrams where used in documenting and detailing the different components and their interactions. From the final design, a similar iterative approach was taken for the final implementation. The first steps, including the Rev0, was seen as a prototyping stage where the team made the basic infrastructure and user interface that could later be shaped in different ways. A similar iterative approach was taken with the client where the prototypes where refined and presented in a biweekly manner to clearly highlight what has been completed and what is left to implement. Toward the end, most members focused on rigorous testing to ensure the validity of the implementation.

Victor: Arriving at our final design and implementation involved a series of steps loosely following the waterfall method. First, we gathered requirements from the clients, Dr. Yang and her grad students, which allowed us to create the first few pieces of documentation for our project. Using these initial documents, we created a first prototype: the proof of concept. We then differed from the waterfall method by showing the prototype and the documentation to

the clients and using the feedback gathered to modify the documentation and implementation. This process of using feedback gathered from the client to go back and modify our documentation and implementation was used throughout the project, being used with the SRS and revision 0 prototype, with the final documentation and the final product, and many times in between. This design methodology allowed us to evolve our first designs into a final product that met client expectations and needs.

3 Design Decisions (LO12)

[Reflect and justify your design decisions. How did limitations, assumptions, and constraints influence your decisions? —TPLT]

Adham:

Darren:

David:

Farzad:

Pedram: As a front-facing software project, the design decisions used in MTOBridge were mostly justified through the perspectives of the stakeholders, often being the client and final user. Nevertheless, there were many factors that also helped justify each decision. For starters, most of the decisions were justified considering the implicit or explicit trade-offs that would be involved. For example, certain frameworks offered less robustness but a higher performance which was not ideal for our scenario. At the same time, there were many constraints that we had to adhere. The program had to be portable and installable on any Windows 10 and 11 machine without the requirement for any other software or applications. These constraint also often included the actual limitations of the framework being used which could vary by use case. Finally, the assumptions and the context for this project, specifically the user behavior and future requirements were other influencing factors.

Victor: Many of our design decisions revolved around the implementation and design of the user interface. Because our project was largely visual, the decisions made were based on HCI design principles to ensure that the interface created would be practical to those using it. Another key influence on our design decisions was the end user, civil engineers, which we were able to take into account by gaining feedback from Dr. Yang and her graduate students. These things led to the visual design of our project being quite simple and straightforward, eliminating clutter and reducing possible user confusion. By far the largest limitation to our project was time: while 8 months seems like a lot when the project is just getting started, it passes by very quickly, and it is difficult to implement all of the features originally planned. Our assumptions regarding the design of the project related to civil engineers' knowledge and familiarity with computer interfaces, and were confirmed by Dr. Yang and her team. Finally, the constraints around our project related mostly to the end environment in which the program would be used, Windows 10 and 11, and did not influence our design decisions heavily.

4   Economic Considerations (LO23)

5   Reflection on Project Management (LO24)

5.1   How Does Your Project Management Compare to Your Development Plan

5.1.1   Team Meeting Plan

5.1.2   Team Communication Plan

5.1.3   Team Member Roles

5.1.4   Workflow Plan

5.1.5   Technology

When it comes to technology, we somewhat followed our original development plan. We did use C++ and MATLAB as our coding languages, and did use our IDE's linters as outlined in the dev plan. We did not use clang-tidy, though this only mentioned as a tool we might possibly use. When it comes to testing, we did in fact use Qt Test for our unit testing framework, however we did not use GCOV for code coverage measuring or Valgrind for performance measuring. We ended up not using these tools as we decided they were not necessary. We also did not use GitHub's built in CI/CD, or any CI/CD, as we deemed, because the project was fairly brief, it would not save use enough time to offset the time it would take to implement. As outlined in the development plan, we used the Qt GUI framework for C++, which allowed us to quickly and relatively easily build our front-end program. We did not use Doxygen to generate source code documentation, instead opting for in-depth code comments.

5.2 What Went Well?

Adham:
Darren:
David:
Farzad:
Pedram:
Victor: Our team meeting and team communication plans went very well in my opinion. We kept to our team meeting plan and consistently met every week, allowing us to go over all the progress that's been made since our last meeting in addition to what must still be done. Our team communication plan, aside from the use of GitHub issues which we did not implement, proved very fruitful, allowing for our team to communicate information and needs in between our meetings. These two combined allowed our team to stay in constant communication, meaning that every member was always up to date and no information was being missed.

5.3 What Went Wrong?

Adham:
Darren:
David:
Farzad:
Pedram:
Victor: One of the things that went wrong for our project's management was our workflow plan. While we had initially intended to use a primary production branch in conjunction with feature branches, we ended up simply using one main branch. The reason for this was simplicity: it was much easier to use one main branch for all changes than have to create a feature branch every time a change was to be made, especially with the way in which we implemented the project. This did, however, cause issues for our project. One of these issues was that half-finished features would end up being pushed to the GitHub, where a second teammate would start working on a new feature based on this half-finished feature, and once the first feature is properly finished it invalidates some of the work done by the second teammate.

5.4 What Would you Do Differently Next Time?

Adham:
Darren:
David:
Farzad:

Pedram:

Victor: One of the biggest things I will do differently next time is to actually use GitHub to its full potential. While we used GitHub for version control and to facilitate all working on the project simultaneously, the lack of CI/CD and GitHub issues was sorely felt. CI/CD could have been used to run tests when code was uploaded and would have prevented multiple regression bugs that we encountered. GitHub issues for tracking changes to be made would have significantly improved the final touches and review stages of our project.