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. Modified context and use case diagrams to be higher quality
2. Added an appropriate introduction to the appendix
3. Modified user characteristics to be more clear
4. Modified the mandated constraints to be correct
5. Fix behavioral overview to reflect changes to requirements
6. Add functional requirements discovered from hazard analysis: FR.16-20
7. Modify FR.1 to have better defined fit criterion

8. Modify FR.2, FR.6 to reflect the change in required user input for analysis
9. Remove NFR.14,15, as they are irrelevant/untestable
10. Add extra risk related to changing the changing of the MATLAB code
11. Fixed wording and grammar all over the document, especially with some of the functional requirements to maintain clarity

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

The design was adjusted based on user feedback over the course of the project and for general improvement, particularly in robustness. This included the following:

1. Removed numerous exceptions in favour of better handling and designing to prevent them
2. The solver section was deemed to not need a save/load system due to its simplicity and so removed
3. Implemented loading parameters from report file
4. Added graph for displaying the moment envelope exclusive to one of the solver modes

5. Altering how input fields are made available to the user and validated by the system

The MIS was updated to reflect the above changes to the design. In particular:

1. Removed exceptions that are no longer thrown
2. Revised some method parameters, primarily changing from receiving string and the associated parsing and exceptions to receiving the types that the string was originally converted to
3. Formalized exception conditions for several modules
4. Added state invariants to Platoon Configuration Visualizer and Bridge Configuration Visualizer
5. Updated methods, particularly for Platoon Configuration Data Format, Bridge Configuration Data Format, Platoon Configuration Visualizer, and Bridge Configuration Visualizer
6. Solver Configuration Loader replaced with Output Report Loader
7. Corrected some grammar and formatting issues

The MG Document was updated to reflect the above changes to the design. In particular:

1. Corrected grammar, tone and spelling throughout the document
2. Expanded the acronym and abbreviation table including the program name and description
3. Revised some of the titles for the modules
4. Clarified some of the secrets for the modules
5. Added links to the traceability matrix for easier use

The System Design Document was updated to reflect the above changes to the design. In particular:

1. Corrected grammar, tone and spelling throughout the document
2. Added an acronym and abbreviation table including the program name and description
3. Revised the Project Timeline to reflect the actual dates that the team was able to meet
4. Removed the "Code Integrartion Testing" phase as the testing was mainly done in accordance to the V&V plan in a later phase

1.3 VnV Plan and Report

1.3.1 VnV Plan

There was quite a bit to change in response to TA Feedback for the VnV Plan, and this was the sole source of implemented feedback for this document. The changes are as follows:

1. Fixed some grammar issues and conversational tone in earlier parts of the document
2. Added reasoning for each linked document as to why it is relevant to the VnV Plan in Section 3.3
3. Assigned specific team members to specific reviews in Section 4
4. Added a pool of predetermined test objects for the system tests and referenced them in input data of system tests to provide specific test data instead of just "valid bridge length", for example.
5. The feedback about some requirement fit criteria, such as the one for FR2, not being fully covered is an issue with the SRS being out of date, not the VnV plan, so that problem was fixed upstream and not implemented in the VnV Plan.
6. Made certain tests, such as NFR10.ST1, less vague.
7. Made it more clear that any subset of the information being invalid is an overall invalid input for FR2 ST2 and FR6 ST2
8. Made it more clear what a "correct value" is for FR2 ST1 and FR6 ST2
9. Changed input decisions to not mention "Text input" to avoid making desing decisions in the VnV plan
10. Added 4 requirements for thread safety and error handling from the Hazard analysis and added system tests for them, to cover scenarios such as a user trying to run the second solver without specifying a discretization length, and more.
11. split up STs for FR13 to be more specific
12. NFR 14 and 15 were removed from the SRS, as they were unnecessary and either not testable or not system requirements. So the issue of no tests for them also goes away

1.3.2 VnV Report

There were some changes to the VnV report in response to TA and peer evaluation. The changes are as follows:

1. More formal word choice was substituted in some instances.

2. System tests were ran again and results were updated with all except for one passing.

3. Unit tests were ran again and the results were updated. More tests passed since rev 0, however there were a few which failed without any implication on the program's functionality at the system level since the UI doesn't allow the provision of invalid inputs.

4. The results have been color coded to make it easier for the reader to identify them.

5. The total number of civil engineers who were subjected to non-functional testing was specified.

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: Honestly I think we did a pretty decent job of requirements elicitation early on in the project, so there were no huge left turns we had to make in the design process. The FRs and NFRs we defined in the SRS and HA back in September and October remained almost entirely static for the duration of the project Most of the evolution was not the design as a whole changing or evolving in significant ways, but rather us refining our understanding of our initial set of requirements and how best to implement them.

Darren: We stayed in regular contact with the clients throughout the project. As we made progress on the design, we reviewed the changes and additions with them. This occasionally raised new insights into what the client wanted and what should be prioritized. Some features were initially requested and then the client determined them unnecessary or another method preferred to fulfill the underlying requirement. The first version was mainly to accomplish the core features of the design. After this, the meetings focused on refinement, particularly focusing on robustness as well as user experience and flow of use, and confirming our decisions with feedback from the clients. The final version turned out much easier to use and more presentable.

David: The design evolved quite a bit. Since we were constantly meeting with Dr. Yang and her team, we were discovering that new design decisions had to be made in order to accommodate their preferences and requirements. We had made some assumptions for the first version that were not true, one of which was that there were would only be 2 graphs that are needed, which changed for our final version. We were constantly discussing changes that should be made, especially in terms of usability. We paired this with with the feedback we received to ensure we were doing the best we could.

Farzad: Our team initially followed a waterfall approach to gather and document requirements for our project. However, as we progressed, we realized the need for a more flexible and iterative approach to deliver a high-quality product

that meets the evolving needs of our stakeholders. We transitioned to more agile-like practices that involved building, measuring, rebuilding, and remeasuring to continuously improve our design and implementation. This allowed us to be more responsive to changing requirements, incorporate feedback from users, and quickly adapt to emerging challenges. Our final design and implementation were a result of this agile approach, which enabled us to deliver a product that met the needs of our stakeholders while ensuring quality and efficiency.

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: I think the main limitation, as might be unsurprising, that defined the project was a limitation of time. The main design decisions that need to be

justified, in my opinion, are the ones that involved omission of features, rather than their inclusion. The main one that comes to mind right now is the fact that the program is left without an installer. detailed installation instructions were created and we tested that non computer inclined folks could follow them successfully with Dr Yang and her grad students, but this is obviously not ideal compared to an installer that just does the more confusing stuff, such as setting environment variables, for you. The installer had to get axed in favor of what we decided to be more pressing functionalities based on our discussions with Dr Yang, but it still would've been nice to have.

Darren: The design was ultimately for a user interface and so was approached with that as the main focus of our efforts. A particular assumption discussed in meetings with the client was that the user would be experienced and familiar with the terminology used, so we limited the resources we used on explaining those. We determined late in the course that this wasn't infallible, as certain terms like "headway" may have a discrepancy between the user perception (distance from the back of the leading truck to the front of the truck behind it) and its intended meaning (distance from the leading truck's rearmost axle to the next truck's front axle). Beyond this, we justified our decisions on information to include, parameters to hide from the user, and how inputs are afforded by reviewing them with the client. We used Dr. Yang and her graduate students to judge its usability and clarity of information - one limitation is that we weren't able to contact an MTO engineer to seek their feedback on the project.

David: I think generally we made the best design decisions that we could've at the time of implementation. There are some decisions that we actually never made, like whether or not to include a menu bar, context menu etc. that are important for any GUI, however we were limited in our time and scope. We were also limited in terms of using MATLAB, which is something that we were not too happy about. We also made assumptions that didn't turn out to be true, which slowed down our implementation since we had to go back and redesign features.

Farzad: Constraints and limitations have played a significant role in shaping our design and implementation decisions. For instance, we were informed that bridge calculations were performed on a Matlab engine, which meant that our application had to implement thread safety to avoid any data corruption or race conditions. Additionally, we had to ensure a strict UI design, as retrieving error messages from the Matlab engine would have been a complex and time-consuming process. Time and development power were also major limitations that affected our implementation choices. Due to these constraints, we had to restrict ourselves to 2D and single angle visualizations, which may have limited the overall functionality and user experience of our application.

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)

As this is a bespoke product created for internal use at Dr Yang's lab as part of a research project, Economic and for profit considerations are not really viewed as relevant by the team.

5 Reflection on Project Management (LO24)

[This question focuses on processes and tools used for project management. —TPLT]

5.1 How Does Your Project Management Compare to Your Development Plan

[Did you follow your Development plan, with respect to the team meeting plan, team communication plan, team member roles and workflow plan. Did you use the technology you planned on using? —TPLT]

5.1.1 Team Meeting Plan

For the most part, we stuck to our team meeting plans perfectly, even over reading week. We did eventually switch to primarily virtual instead of in-person meetings, and the meeting time changed from 5:30 on tuesdays to 6pm on thursdays once second semester hit and everyone's schedules changed, but there were no real deviations otherwise.

5.1.2 Team Communication Plan

A few more deviations here, namely, Git issues were not used by the team, and the primary communication for action items became the discord group and meetings. We also met with Dr Yang in person instead of over zoom, thankfully.

5.1.3 Team Member Roles

Considering how little we knew about what we were getting into, this section is also quite accurate. Adham did in fact act as lead and chair meetings, Darren facilitated the hybrid meetings with Dr Yang when not everyone could make it to the room, and David was the technical lead for the implementation, Victor was also given a more complex part of the implementation due to his experience in C++. The only deviations are really that Adham ended up being the primary point of contact with Dr Yang over the course of the year, and Victor shared the role of Github maintainer with Pedram.

5.1.4 Workflow Plan

This was the one we were most off the mark on. We did not have separate dev and prod branches, we did not implement CI/CD, and we did not use git issues. Basically not a single part of this Workflow plan was followed. Oh well, it worked out well enough.

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?

[What went well for your project management in terms of processes and technology? —TPLT]

Adham: I think we did very well in terms of organizing ourselves, meeting consistently, identifying action items, assigning work fairly, and getting things done in a timely manner. We never missed a deadline, even when things got very busy, and had very few issues in terms of team communication or missed meetings.

Darren: The team had very effective communication and division of work. We had meetings within in the team at least once per week and met with the client every other week. We used our team meetings well in discussing and dividing tasks, as well as following up on assigned work and assisting each other in the case of blockers. Client meetings were used to confirm the direction of implementation and that the product being developed was what the client wanted. I felt the team also had a very good atmosphere and sense of cohesion. Particularly, there was no tension such as a teammate slacking on their part or failing to contribute meaningfully. Communication was open and comfortable.

David: I think generally everything went well, especially our communication, meetings, and our teamwork. We were very open to each others' ideas, happy to receive feedback and everyone got their work done on time. We were all also willing to help each other if need be. I think our communication with our client went well as well, as we met with Dr. Yang and her team consistently, and we accepted valuable feedback from them as well.

Farzad: In terms of project management, several aspects of our project went well. Firstly, we established clear and achievable project goals and objectives at the outset, which helped us stay focused and motivated throughout the project lifecycle.Regular team meetings and communication channels were established to ensure that everyone was on the same page and any issues were addressed promptly.

Pedram: The project was a complete success and as such I believe there were many things that went well during the term. Most importantly, the team had a very welcoming dynamic where each person was helped and encouraged to achieve more, creating a positive feedback loop. We all knew our own skillsets as well as limitations, making the division of work often very easy. We were also very understanding of each other's situations and showed the utmost flexibility toward each member. Furthermore, all team members had dabbled with the programming language used for the project, C++, which was very good for bootstrapping the early works. The regular meetings as well as the client meetings were both also great assets in managing the process. Lastly, our team was lucky to have a member (Adam) to be enthusiastic about running and keeping the meetings on time, which was very nice. This could be a suggestion for all the groups in the future.

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?

[What went wrong in terms of processes and technology? —TPLT]

Adham: We made basically no use of git and its features outside of using as a code repository. We didnt really stick to our workflow plan in the slightest, and that definitely hurt us sometimes in the implementation, where someone would push code that didnt compile and stop the workflow until it was fixed, or people would step on each others toes when it came to implementing features or solving problems sometimes.

Darren: Git wasn't used to its full potential, as we all worked on the same branch. This saved on time by cutting out the process of switching branches and any potential confusion or dropped features due to some code being left behind on an unmerged branch. This also caused some discrepancies in team meetings where a bug appears on the build used when reviewing with clients while that same bug has been fixed in a commit overnight, leading to repeated work or extra time spent confirming with teammates that that is indeed the same bug and whether it is already fixed. There were also initial difficulties getting the project set up on everyone's computers due to dependencies such as the MATLAB installation being in different directories, but the files required for changing them being synced by Git.

David: There were a lot of things we wanted to do in terms of setting up our project and our Github, but we just never got around to it. We also ended up finishing deliverables the day they were due very often, just like this very deliverable. I think we didn't really live up to the first deliverable where we said we would set internal deadlines, and use all the features of Github and other tools like Valgrind to help us achieve our objective, which was to make a usable program.

Farzad: In terms of technology, one issue that we faced was not utilizing Git and its features like pull requests effectively. This led to a lack of coordination and version control, resulting in some confusion and delays in the development process. As a result, we had to spend more time resolving merge conflicts and ensuring that all team members were working with the latest code. On the project management front, one issue that we faced was finishing deliverables on the day they were due. This put unnecessary pressure on the team and made it difficult to address any last-minute issues that arose.

Pedram: The project was definitely full of challenges and things that simply went wrong nevertheless, we quickly found the key is to be adaptable and

creative. One of the biggest challenges that we faced was the limitations of the framework we chose to work with, Qt. Although a very popular and powerful way of developing windows applications, it can have a steep learning curve. For our application, where the UI was one of the most important aspects, other frameworks more focused on GUI could be considered. For our technology, there was a considerable amount of time spent on creating the infrastructure of the program. Lastly, the GitHub might have not been used to its full potential. The team did lack the continuous integration and testing operations along with the proper use of issue tracking.

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?

[What will you do differently for your next project? —TPLT]

Adham: Besides the obvious "Use Github More Effectively" I feel that personally as team lead I could have often done a much better job of chairing the meetings. Things such as letting people know the agenda for each meeting ahead of time, and more clearly communicating the actions items at the end of the meetings. I also tend to drone on often, which I could definitely tell led to me losing people's attention often, which would subsequently lead to important information being missed. To sum it up I think I have a lot of work to do when it comes to clear and concise communication in project management.

Darren: I would work on separate branches as we originally stated. I would also want to investigate means for better task and Git tracking beyond GitHub itself for better accessibility and organization. There were times when some confusion arose on remaining tasks due to almost all team communication being conducted in the same single text channel. While this was usually handled well by simply asking around or addressed in meeting, it could be improved or made more accessible nonetheless with tools such as Trello.

David: There is not much I would do differently, I really enjoyed working on this project. I do wish, as I mentioned earlier, that we followed though with some of the decisions in regards to CI/CD, because it would have made our lives much easier in the long run. I also wish we made separate branches instead of pushing straight to main, since that caused issues occasionally. I would have also wanted to push for a meeting with the MTO so that we could receive valuable

usability data, as well as general suggestions.

Farzad: This project was a pleasant experience, primarily due to the team members' contribution. There are very few aspects I would alter if I were to commence it again. However, focusing less on the revision 0 documentation (given there was a revision 1) and starting implementation sooner would have been advantageous to produce a superior final product. By doing so, we could have received more feedback and transformed our project management approach to adopt a more agile style.

Pedram: In all honesty, I found my capstone team and project very interesting, and I am truly happy to have had such an opportunity. Nevertheless, if I had to redo a new capstone, with similarly great team members, I would encourage my team to spend much more time in understanding our true capabilities as well as the tools and technologies that would be best suited for our needs. In more detail, by spending more time on the proposed solutions, we could have possibly come up with a hybrid set of tools that is more suitable for our goals and easier to implement. At the same time, I would have tried to scale our user testing to get input from a much larger set of audience, some of which could actually be Civil Engineers working in MTO (for our case specifically).

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.