Final Team Reflection

Customer Value and Scope

The chosen scope of the application under development including the priority of features and for whom you are creating value:

A: The chosen scope for our UML-program was to make a small, yet polished program. Our PO was the current examiner for the OOP course at Chalmers (TDA552). Therefore we prioritised making a program that was easy to use for students that were not familiar with UML diagrams, specifically class diagrams. To make the program easy to use for beginners we made a menu system that reduced the amount of prior knowledge required to make a class diagram. For example attribute visibility signs and arrow types are dropdown choices so the student doesn't need to know what the arrows look like, and can instead learn by doing. We also included tooltips and automated features such as pathfinding for arrows and a grid to help line up the diagram to make it neat.

Our PO also wanted a program that was easy to use, but also quick enough to enable easy sketch up of diagrams. Supporting features for this were pathfinding for the arrows, a system able to edit the diagram in a text file, and a grid that automatically lined up the classes to reduce time spent “fiddling” as well as generally decreasing the amount of clicks needed to complete an action. One challenge we faced when designing the program was to balance the “easy to use” and “quick to use” parts of the program. Nevertheless, with the guide of our PO we believe that our program matched the scope.

B: Although our scope with the project was to make a small polished program we had a lot of features we wanted to implement at the beginning of the project. A majority of these had to be cut off later on in the project. Therefore, even if we already had worked on some of the design parts for those features, we could not use them. Even if our PO approved these features early on he warned us that it might be too much for us. If we as a group had listened to that warning at the beginning of the project it might have saved us some of the effort in designing those parts. This is something that we want to learn from in the future.

Furthermore, we would like to have defined the scope of our project more precisely. For example, we stated that our program should be small and polished. Although reasonable we realised whilst working on the program that this goal is not well defined. If we had defined “small polished program” clearer these extra features would probably not have been worked on in the first place.

A → B: One approach to achieve this might be to use several smaller milestones that we focus on depending on the phase of the project. For example, in the beginning of the project our first small milestone was actually a small program with all the core functionalities, we should have continued this workflow throughout the project. As the project progresses the scope would orient more towards the final scope, which is the full polished program.

The success criteria for the team in terms of what you want to achieve within the project (this can include the application, but also your learning outcomes, your teamwork, or your effort):

A: Our success criterias to determine if we had achieved what we had established at the start of the project were:

- Being able to present a complete application after our last sprint
- Learning to work as a team with the principles of scrum
- Learning to use agile work methods and utilizing the members different strength and experiences

These criteria have not changed during the course of the project.

B: In hindsight, similarly to above, we should have had a better definition of the completed project. Also, since our program's repository is on git we would have liked to add “becoming more comfortable working with git” to our success criteria.

A → B: By defining the success criterias in greater detail throughout the project we would probably have the wherewithal to better define whether we achieved them in the end or not.

Your user stories in terms of using a standard pattern, acceptance criteria, task breakdown and effort estimation and how this influenced the way you worked and created value:

A: To achieve our goal of creating a UML application we had to break it down into several different user stories that followed a standard pattern with tasks and time estimations. By having a time estimation on each user story, we gained valuable insights on the expected workload. This made it easier for us to balance out the work between the members during each sprint. The usage of tasks in each user story was beneficial in that it both provided more concrete implementation details and enabled the rest of the team to follow the progression during each sprint.

In general, breaking down the application into smaller goals (user stories) gave us an idea of how much we would be able to implement over the given sprint. Moreover, using user stories when discussing the project with the PO was an effective way for him to understand our plan for each week. Finally, this method of dividing the application into user stories was a great way to get an overview of what we will be able to finish.

B: An improvement we believe would help us mark whether a user story is completed/uncompleted is to have better defined/more concrete acceptance criterias. Even though we did not encounter any serious miscommunications whether a user story was complete or not, it would have been easier to get a grip of the project status if they were clearer.

As mentioned in part A, the user stories were an effective tool for not only structuring and organizing the project but also getting an overview of the progress. However, one improvement for the future is to keep the user stories as updated as possible, for instance checking the completed tasks. In this way, other group members can easily see and follow the other team members' real time work progress.

A → B: By adding more concrete acceptance criterias to the user stories, the first point could be realised. The second improvement can be realised by adding a requirement to the group contract to always keep the user stories you are responsible for updated.

Your acceptance tests, such as how they were performed, with whom, and which value they provided for you and the other stakeholders:

A: Our Definition of done used for our acceptance tests was that:

- Each task in the user story is deemed completed by the member responsible
- One of the Git Guru thinks that the code looks fine
- Robin, our PO, accepts the results

The acceptance test regarding our PO was executed by demonstrating the changes to him via Zoom as the last thing we did in our sprint. By showing him the current state of our application, we could choose what to show and ask feedback on and what to skip, rather than having him click his way through the application. This was both time effective and valuable for us and the PO.

B: To have more defined, accessible and specific acceptance tests so that it brings value into our work progress.

A → B: A possible way to reduce the gap between A and B is to firstly rethink our Definition of Done so that it becomes more specific and brings more value to our work. In that way, we believe that the usage of this could have brought the whole team more understanding and knowledge on every user stories' progress and not only the responsible one. The Definition of Done was in general vaguely defined resulting in that even though we had acceptance tests, the usage of them could have been more direct. All of our acceptance tests were already necessary steps to take in order to continue our work.

Also, our tasks in each user story could have been explained in more detail and divided into smaller tasks so the rest of the group could follow the progress more closely. Finally, the acceptance tests could have been written not only into each user story but also in our group contract to make the acceptance tests more visible and accessible instead of having the criterias written down in one of our meeting protocols.

The three KPIs you use for monitoring your progress and how you use them to improve your process:

A: Our chosen KPIs were:

- The ratio of planned to spent time
- The general stress level of the group
- How much the group learned each sprint

The ratio of planned to used time helped us both to keep track of the workload amongst the group and gave us data on how many work-hours each sprint had. This could with the stress level give an indicator whether the planned time each was reasonable. The stress level could also give an indicator of the group's overall well-being. Finally the KPI “How much the group learned each sprint” could give an indicator whether the team member was sufficiently challenged during each sprint.

B: Perhaps we could have had more “hard” values as KPIs, that might have led to a better understanding of the project in its entirety. Hard values are also quite easy to keep track of and update if you implement a good system.

We could have logged our time for individual tasks or user stories. This would have helped us get a better idea of how long each user story took to complete compared to how much time we planned for it. This would probably have helped us plan our time better during the scrum poker sessions.

We should probably have spent more time reflecting on our week's KPI’s. Even if we feel we would have reacted if the stress level of the group was high for example, we did not have a specific time when we sat down and discussed it.

A → B: We could have dedicated some time every week to reflect on our KPI’s, and we could have implemented some other or better KPI’s at the start of the project, but we could also have updated our KPI’s during the project.

Social Contract and Effort

Your social contract, i.e., the rules that define how you work together as a team, how it influenced your work, and how it evolved during the project (this means, of course, you should create one in the first week and continuously update it when the need arrives):

A: The social contract was rather extensive and included, among other things:

- The goal of the project
- Definitions of the different roles in the group
- Rules and guidelines for meetings
- Work principles
- Channels of communication
- Documentation handling
- Github rules
- Rules regarding decision making
- Guidelines for handling conflicts in the group
- KPI:s
- Various deadlines for the project.

We are mostly content with what was defined in the social contract, even if some parts were not needed. However, we still regard these as necessary parts for a social contract since they could be needed (for example "guidelines for handling conflicts") in future projects. Furthermore, the social contract was modified during the project. This was mainly when new information came to light that could improve the contract.

B: That being said, updating the contract was mainly done in the early to mid stages of the project. Had we continued with this throughout the project our social contract would presumably have been even better. Another consequence of not updating the contract is that the social contract isn't a precise replica of reality as for example roles shifted during the project that were not updated in the project. This is something that we want to improve on in the future.

A → B: Updating the social contract usually fell behind since our focus was mainly on the program and how to complete it within the given timeframe. As stated above this became more prominent in the later stages. To resolve this we could have added a point to the meeting agenda to ensure that the social contract was up to date.

The time you have spent on the course and how it relates to what you delivered (so keep track of your hours so you can describe the current situation):

A: According to the time KPI our time spent to what we estimated was rather accurate. This is not necessarily a good thing, considering that we had consistent issues of combining our git-branches separated work into one single project and we were not always able to properly finish/debug our user stories, which was not included in our time estimation. Both of which led to a kind of “time-debt” that had to be repaid at the start of the following sprint.

B: In the future, it would be good to include the time spent merging the project together at the end of a sprint as part of our time estimates. This would allow us to more accurately gauge how much time would be required.

A → B: In the future we should take time to reflect on how much time we spend on activities that are not currently included in our time estimation. These things need to be accounted for to reduce the “time-debt”.

Another tool that would really help us with keeping track of the time spent on meetings and merging as well as help us keep track of upcoming meetings and project activities is the use of a project calendar. This would allow us to write the estimated time for meetings keeping track of when we have the meetings and we could after the meeting adjust the time of the meeting to the actual time.

Design decisions and product structure

How your design decisions (e.g., choice of APIs, architecture patterns, behaviour) support customer value:

A: We intended to make the program modular and structured so it could be expanded upon. The end result is a bit messy, but functional. We decided to use the MVVM pattern as a core but what we ended up with (probably) isn't really MVVM but more of a MVC pattern. This didn't really become a problem because MVC did also suit our application.

B: We should have had a stronger foundation that eliminates uncertainties about communication between the frontend and backend. The program is quite modular, but there is room for improvement. We could also have used more proper OOP patterns in the frontend.

A → B: We could have spent more time planning the foundation of the program together as a group, and most likely the whole group together (instead of backend and frontend working separately). This would also lead to better cohesion in the program.

Which technical documentation you use and why (e.g. use cases, interaction diagrams, class diagrams, domain models or component diagrams, text documents):

A: We have used JavaDoc in the project and it has helped us, though there is a lot of undocumented code. Early in the project we had a good UML diagram during the planning stages. We also used a sequence diagram to help determine how the backend and frontend communicated.

B: In the end we were not very successful with JavaDoc, which would probably have helped us keep track of the project as well as making life easier for maintaining or updating the application in the future. Keeping the UML and JavaDoc up to date would also have been useful during the project.

A → B: We could have kept a slower pace and actually demanded proper documentation at the end of each sprint to consider the code done. This is obviously hard in a rapidly changing codebase but the actual time needed to document your code is a fraction of the time it takes to write it.

How you use and update your documentation throughout the sprints:

A: There were a mixture of approaches to this during the sprints. Some parts of the code had documentation and some did not. With exceptions to the facade classes that were supposed to act as the middleman between front and backend the documentation was largely lacking. During later parts of the project two individuals could not contribute as much code wise, due to lacking competences and time issues. They were thus tasked with documenting the backend code to the best of their ability. However this entire approach is neither efficient nor the most effective.

B: The perfect scenario would be that the documentation on the code is written in parallel with the actual code. This would ensure that the writer of the code would be the same as the one documenting, which would ensure better precision of the documentation aswells as probably take less time

A → B: To achieve this the social contract would have to stipulate some clause to ensure that everyone had to be more diligent with documentation as well as some sort of way to review this.

How you ensure code quality and enforce coding standards:

A: Two members were responsible for our git repository. They were also responsible for going through and checking pull requests to our production bransch. To make sure that we all followed the same coding standards they also wrote guidelines for our code in the README in the beginning of the project.

We also discussed a lot of coding patterns and standards we used in our group meetings to make sure that our code followed a solid design. Lastly we had two members that during some parts of the project focused on going through and documenting parts of the code.

B: Even if we had some methods for enforcing good code quality and coding standards we could definitely improve on this in the future. There was especially some lack of code quality in the later parts of the project where we had more focus on getting everything to work properly rather than enforcing good coding practices. In future projects we would like to keep the code quality throughout the entirety of the project.

Discussing the code in group meetings worked to make sure that we followed design patterns properly and that we had a good code quality, even if it sometimes took up a lot of time in our meetings. We believe that discussing code is an important part of keeping good code quality but in the future we would do that in special code discussion meetings.

A → B: To ensure better code quality throughout the project we think that we would need to be harsher on the code we write ourselves as well as the code we review. This might take up extra time, because more code would have to be rewritten. However, we believe that in the long run this would save time because of the better code quality.

Application of Scrum

The roles you have used within the team and their impact on your work:

A: The roles we had in the group were:

- Scrummaster (2 people)
    - Focusing on administration
    - Looking through and taking care of the scrum board.
    - Keeping more of a complete picture of the program
    - Making sure the group can communicate with each other.
- Git-guru (2 people)
    - Reviewing pull-requests
    - Helping with git-related problems
    - Responsible for the git README and rules about code commenting.
- Backend (4 people)
    - Responsible for the technical model of the program.
- Frontend (3 people)
    - Responsible for the graphical parts of the program
    - Responsible for the controllers that connect the graphics to the model

Because there were 7 people in the group, some people had 2 roles.

B: During the last week of the project we realised that it was more efficient to divide groups into pairs of 2 or 3 depending on what they are working on instead of backend and frontend. This resulted in a combination of front and backend individuals in most groups which greatly improved the communication between the before largely separated groups. This in turn increased the efficiency and enabled trello cards to be completed in their entirety instead of having some small criteria (usually the connection between the graphics and the model) remain unfinished.

A → B: In the future we would still keep the division of backend and frontend, however we would at the start of every sprint divide us into smaller groups where both front and backend individuals could cooperate, although this would of course vary depending on the user story. It would not always be necessary to keep the same smaller groups each week, even if it would be beneficial in some cases such as if the user story is still not finished.

The agile practices you have used and their impact on your work:

A: Our work method is based on the Scrum methodology which is an Agile approach. By applying the Scrum principles, it has been helpful in terms of structuring a project and delegating responsibility and tasks among a team of seven members. The impact of the Scrum method has been huge since we have been applying many of the principles, for instance working in sprints, using user stories, having specific roles such as Scrum Master and Product Owner etc.

Also, we performed time estimations of our chosen user stories at the start of each sprint, allowing us to have a better understanding of how much effort would actually be required. On occasion, we would use these time estimates to reprioritise our user stories if we felt that there would be too much work to be done during a sprint overall or if any team members would have much more to do than other members.

In general, we have taken an iterative approach towards our project through our sprints. Moreover, we reflected on our progress and teamwork every week.

B: One Agile process aspect that could have been improved is creating the user stories. They could have been more defined and split down into smaller pieces. Another aspect that could have been improved regarding user stories is that the user stories should be based on the end users expectations, instead, we implemented what we thought was valuable to include. Because of the user stories not being great we sometimes had a hard time dividing down the user stories into chunks that we could finish in one week. This is definitely something that we want to improve in our next projects.

A → B: One tool to help us evaluate the program and get better user stories is to perform user tests to gain insight of the user requirements. That would allow us to adapt to their requirements on the program and then create user stories. We could also incorporate the user test to our definition of done.

The sprint review and how it relates to your scope and customer value (Did you have a PO, if yes, who?, if no, how did you carry out the review? Did the review result in a re-prioritisation of user stories? How did the reviews relate to your DoD? Did the feedback change your way of working?):

A: We did have a PO as mentioned earlier, and his feedback was important to us and how we prioritize our user stories as well as how features were implemented. His approval of a feature was a part of our DoD. His feedback changed how we worked and what we focused on depending on what features were important to him as a client.

At one point when we asked for his approval of the functionality of some features he suggested that we take our time to polish the product instead of adding new features for next week's sprint since he didn’t approve of the quality of the work we had produced at that point. At that point we spent a full week refactoring and polishing our work.

B: Since Robin was probably very aware of how a school project works, he was probably more relaxed when we showed him buggy code during the reviews. Another project owner would maybe not be as pleased with the state of the product during those reviews. Overall our cooperation with our PO worked very well.

A → B: To avoid a potentially unhappy customer as well as keep the project under control we should have been completely done with each user story before we showed the PO. This would allow us to complete a user story, and actually consider it finished instead of constantly having work to do on the previous sprint.

Best practices for learning and using new tools and technologies (IDEs, version control, scrum boards etc.; do not only describe which tools you used but focus on how you developed the expertise to use them):

A: During the project we often relied on each other when learning new things. For example, we had a github meeting in the beginning of the project where the members with more github experience explained parts of github for the members with less github experience. We also discussed a lot in the group about tools and coding techniques that we were unsure how to use.

B: One problem with learning from each other is that it relies on that someone in the group having experience and knowledge about the subject. In the future we would like to get rid of this weakness by not only relying on each other for learning new things but also using methods to learn things that no one is familiar with together.

A → B: If we as a whole group, or at least several members of the group, needed to learn a new thing it would take a lot of time and effort if each and every one of us needed to do the research and learning on our own. Especially researching a field that you are unfamiliar with can take a lot of time. One method to reduce this time spent is to assign only one or two members for doing the research and then after that letting those people teach the rest of the group. This would probably take less time than every member doing their own research because one thing that we learnt during this project is that teaching each other is a time efficient way of learning.

Relation to literature and guest lectures (how do your reflections relate to what others have to say?):

A: This is not applicable for our project since we neither had guest lectures nor literature

B: Having a guest lecture with someone with work experience in the Scrum methods would have been interesting listening to. We believe that hearing about real life examples about Scrum and the advantages of it would have been valuable for us to apply the principle of Scrum more intentionally.

A → B: By suggesting to the examiner to invite a person with scrum experience would have realised our desire.