Team Reflection W9

Customer Value and Scope

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

  During this sprint, the priority of features have become more definite and we have focused on doing what is most important rather than trying to do many new things that are more of additional features. We have focused on finishing the largests user stories in order to create value to our stakeholder and decided to not implement others. This is the result of either too optimistic planning when it comes to what was doable during our development time or/and too low productivity in the team. An idea to potentially fix this would have been some sort of burn-down chart to constantly track how close the team is to completion of the app as well as spending more time on planning our work, we think this could have led to increased productivity.

- **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)**

  The success criteria has not changed.

- **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**

  We noticed this week that we had done effort estimation of tasks but not on user stories. This is bad because it makes the team think more horizontally instead of vertically since finishing tasks gets priority over finishing user stories. To address this the effort estimation would be done on user stories to encourage vertical thinking.

  In general the group has thought too little about “slicing the cake”, even though it has been discussed it is still something that could have been addressed more. This was most apparent early on but is still a problem. Now it is quite hard to do something about it but we believe that a focus on effort estimation for the stories would have made a difference. Especially if we would have planned and made estimations earlier on in the project, we always just did them at the start of every sprint/week and didn’t put enough time into looking into how much work each part actually required. Since we knew what we wanted in the first week we could’ve spent time looking into those things then, along with needed APIs. This would have led to better planning, which means we could’ve been more effective in how we divided workload and have more clear user stories and tasks early on.

  A bigger focus on delivering demo functionality every sprint is also something that could be improved. This has largely worked well during our development but some

features of the program were left half done instead of finished for quite a while longer than wanted. This would not be solved, but at least helped with a bigger focus on verticality.

Another thing we have had problems with but got quite evident during the last sprint is the problem related to how we do task breakdown. Generally we have been quite good at breaking user stories down into small tasks but we have not been as good at figuring out the number of tasks needed. Often we realize while coding that more things need to be done than we planned. This caused miscommunication in the group regarding what was worked on and also made user stories take longer time than expected. This also made delegating more difficult since our task breakdown often was incomplete and led to some features taking more or less time than expected. This, we believe, could be solved by doing two things. One: Spending more time on task breakdown to make sure that we understand the demands of the user stories better. Also breaking down the tasks into smaller tasks would have helped us with estimating the size of the stories. This is something that could be done more individually since it would not need 7 people working on it at the same time and the meetings would stay relevant for the whole group. Two: This one is harder to do but a greater experience in the team regarding the Android framework would have made sure that user stories got broken down into tasks in a more complete manner than they did before. Of course members new to Android Studio couldn't have had experience from day one as they were new to it, but we could've been more effective in getting everyone started in learning it.

In general our task breakdown has improved a little last sprint since a list was compiled with all the features that were left in the program. The effort estimation also went better, we think it has to do with that everyone familiarized themselves more with the task breakdown.

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

  Same as last week but it is worth noting that we misunderstood the term acceptance test from the start, which is obvious when reading earlier Team Reflections. This is nothing that has affected our process we believe since we did, what we now understand is acceptance testing, anyway.

- **the three KPIs you use for monitoring your progress and how you use them to improve your process**

  We didn't change the KPIs for last week as it felt like it was too late for it, so the same flaws as we had earlier remain. Even though we didn't change them we've reflected on what we could've done to improve our APIs, but to get anything out of those changes they would've needed to be implemented way earlier on.

  What change we suggested last week still remains, which was to keep track of how many of our planned tasks to get done actually got finished. This can be improved on even further by keeping better track of our progression of user stories as well, how

many substories we planned on getting done, and how close we are to finishing the user story as a whole. We've noticed that we have had bad estimations for how much work is needed for each part, and that is partly understandable as neither of us had worked with certain parts ever before. But keeping track of how the progression is going would've given us a better idea of how hard other substories related to the same epic would be, and how much time and how many people we would need to work on it. Hopefully this could have also made breaking down tasks easier, but it's hard to know without trying out having it implemented.

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) There is a survey you can use for evaluating how the team is perceiving the process and if it is used by several teams it will also help you to assess if your team is following a general pattern or not.**

  As in previous weeks our social contract hasn't been updated.

  We've noticed that some team members have spent significantly less time on helping with preparing documentation such as the team reflection for meetings throughout the project. Most of us have missed out on a week or two, but this is an overarching thing. This was first brought up when going through what was written together last week, but wasn't written down as we seemed to forget to do so after bringing it up. We have earlier agreed to split this workload evenly, but it seems like some have had to spend extra time preparing which of course results in those members having less time to code. If this was noticed earlier we could've made sure that everyone was doing their part in the documentation, which would have resulted in everyone having about as much time on coding.

  In our social contract we say that we should code in pairs to be more efficient and get stuck less. We haven't really followed this and this is something that could be improved. To make sure that we follow the contract and code in pairs we could have planned more and more thoroughly.

- **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)**

  We can see a clear pattern that more concrete work has been done in a shorter amount of time the further into the project we have come. This is no exception this week.

  The first weeks we spent a lot of time during meetings, understanding the techniques of Scrum and for the ones of us that had no experience in Android Studio - to learn

the program. Now, everyone has improved their coding skills which is clearly noticeable as everyone can handle things themselves in a more efficient way.

We have focused less on time spent in general this sprint in favour of actual task completion, which has worked fine and made everyone work for the tasks at hand, instead of hours in the time-log.

Some members have throughout the entire project spent a significant amount of time helping the other members every week. This has hopefully resulted in the process being faster and/or the code getting better for those who needed help, but at the cost of time spent on working on code for those helping.

In the beginning of the course our group had problems with keeping the meetings short and efficient. It has improved a lot since the beginning but it still has potential for improvement. That the meetings sometimes were seen as inefficient and partly irrelevant was brought up after about 3-4 weeks, which is pretty late. If someone had brought it up/noticed it earlier we would have had more time for improvement since you have hard fixing a problem no one knows needs to be fixed. Another way of improving this could have been if the scrum master had a more clear role during meetings. We are a group of motivated students where most of the group has a lot of opinions. This is certainly nothing negative, but sometimes it would have been good with a person who chaired the meetings when we got off topic - suggested the scrum master. If we had done this again we should have had a meeting agenda from the first week and someone who ensured it was followed. Sometimes important things were discussed between only two or three people, these things could be discussed in another meeting that for example could take place after the group meeting.

Design decisions and product structure

- **how your design decisions (e.g., choice of APIs, architecture patterns, behaviour) support customer value**

  We haven't added any new APIs in this last sprint, but in general throughout our project the APIs have definitely simplified and sped up the creation of parts of our program. The cases where we've used APIs are also for core functions we decided we needed very early on, so the need they fulfill is of great value.

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

  This sprint when several different members of the team would use the saving and loading functionality for the first time, there was some confusion about how it worked. This would have been largely solved by having a wiki-page on how to use the system. Instead some members of the team that were more familiar with the systems had to directly help with some of that implementation which took some time. Therefore it is believed by the team that a wiki-page, even though it would have

taken some time to write, would have saved us time in the end. Now there is no real meaning to creating it but it would have been beneficial for our productivity.

- **how you use and update your documentation throughout the sprints**

  Same as before and our class diagram was not updated due to lack of time.

- **how you ensure code quality and enforce coding standards**

  We found that with an increasing amount of features that were not really planned, at least the details of some of them, core functionality of the app would not support these new features. Therefore some quick solutions had to be put in place which made the program considerably less scalable. This was even more noticeable this last sprint. This was the case for us especially with the interaction between the graphing and save functionality. It is too late now to fix this but if we had the chance the first thing to do would be to spend some time before beginning the implementation to map out the program better. This would not only improve the quality of code but also made us more productive since features would be easier to implement. In general though the code that has been committed has been quite bug-free since all model classes with behaviour are unit-tested.

  This week our code review has not been as effective at ensuring code quality as it has used to be. The results of the code reviews have never been definitive, which means that it's up to the individual, whose code is being reviewed, to decide if they want to implement the changes requested. Therefore a lot of requested changes get prioritized in exchange for the last polish of the program. In an ideal world this would not have been the case but since there has been somewhat of a time crunch this sprint it has been hard to avoid.

  Some of the team members felt that due to the workload, code quality has not been prioritised. This could have been avoided by taking on less tasks or better planning from the beginning. The breaking down of tasks could have also improved the code quality since it is then more clear what to do. Experience also has a big part in code quality since some members in the group don't have enough coding experience to write with good quality. This could have been solved with sticking to our contract regarding coding in pairs to maintain the quality throughout the code.

  In general more things should have been done in previous sprints to avoid this time crunch which affects the quality of code. Either productivity wise or getting a better opinion of what is feasible to implement during the time period we had.

Application of Scrum

- **the roles you have used within the team and their impact on your work**

  We have changed scrum master each week to give everyone a chance of trying to have the role. This approach has caused some problems since no individual scrum master will have more than a week's experience. This will of course lead to their

performance being not as fruitful to the team when it comes to removing obstacles, speeding up work process around meetings and documentation, etc. compared to if a more experienced scrum master had the role. We think our choice of changing scrum master every week also affected the overall productivity of the team, given that nobody had a complete overview of the project it was difficult to negotiate with the product owner. This is the first, or maybe second sprint that the team negotiated with the product owner on what was doable. Earlier the team tried to deliver everything the stakeholder was expecting without really negotiating what was possible in the time we had which forced us to reprioritize quite a lot during this last sprint. Also, especially in the beginning, it was a bit unclear what role the scrum master should take. It made more sense for us as we made a document with “definition of roles”, but we still believe that the scrum master could have been assigned more responsibility. Sometimes the scrum master just felt like anybody in the group. We still believe the fact that it gives everyone the chance to try the role outweighs the potential improvements to the project as we prioritize learning and getting experience in this regard.

- **the agile practices you have used and their impact on your work**

  Nothing new.

- **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?)**

  We have improved in efficiency during the sprint reviews. We are now also more comfortable with the reason for having a sprint review every week which makes it a lot easier to understand the benefits of it and use it in future work.

  Our PO has had a clear picture during the last weeks in what is prioritized to finish before the deadline. Our PO has communicated to the rest of the group which has made the planning a lot easier during the last meetings.

  Since we made clear the priority of our tasks and user stories during our last sprint review, we could just follow this but we did decide to leave out some non essential functionality from the project scope due to lack of time and were forced to reprioritize. In the future we will make sure to not promise functionality that can not be finished and we will take with us that it is better to start small, negotiating with the PO, before you know how much work it takes.

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)

Now we learn a lot from each other by helping group members when someone gets stuck. Mainly William and Elias, who has a lot of experience in coding, have helped the rest of the members, which have been very significant for the work efficiency. The plan is that this would continue if the project carried on.

Breaking up user stories and working with sprints have been mostly learnt by experience when working with this group. In future projects we will know the value of breaking up user stories and make sure to allow it time and concentration.

The debugger tool has not been known to all team members and therefore is something that could have been used more. This week two team members used it and got to know about it for the first time. Going through tools like these in the beginning of the project would have improved our work.

We have also spent a lot of time fixing bugs in certains parts of our application, this due to mainly a bad design, we think this could have been avoided if we had spent more time planning in the beginning of this project, we would definitely have chosen a different design for our quiz page.

- **relation to literature and guest lectures (how do your reflections relate to what others have to say?)**

  We feel like this is the first sprint that we fully utilized our supervisor to get feedback on how we are doing. In general we feel like if we had used the supervisors better we would have avoided some of the pitfalls that have been mentioned in our reflections. We also looked into some info about effort estimation which affected our reflection a little.