Final Report - Reflection of the 16 points

Group: Eurythmics
Group members: Jenny Carlsson, Omar Suliman, Eugene Dvoryankov, Ida Nordlund, Desirée Staaf, Oscar Palmqvist & Fabian Flaa

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 overall idea with our project was to create an app where the user can rate films and series. However, the scope of the project did have to change during the project as there was not quite enough time to develop all the functionality which we had intended to. The most major feature which we had to exclude was to include TV-series in the application. Including series would have prevented us from fully implementing and polishing the rating of movie features, which we decided to focus on. The product owner was considered during the whole project, we had weekly demos and considered the feedback when selecting our user stories for the next sprint. We tried to work within the *thin-slice-of-cake* principle, and thus prioritised creating as much of a finished product as possible in each sprint. Each sprint came to focus to some extent on a single page in the application, thus developing both the front end and back end needed for that page in the same week.

Sometimes we became a bit confused about which features we wanted to prioritise and what features the stakeholder wanted to prioritise. This should be fixed, otherwise we risk not considering our stakeholder enough and therefore not focusing on creating value for him. Another thing that we struggled a bit with is the thin-slice-of-cake principle. We tried to focus on a specific page in the app each sprint, but there were some problems with dependencies between different tasks and user stories. This made it hard for everyone to work on their user stories independently and for the team to deliver a finished slice at the end of the sprint.

(B) In a future project we would like to improve our communication with the stakeholder further to make sure that we prioritise the right features throughout the project. We would also like to improve our work with the thin-slice-of-cake principle to help us deliver a valuable product in each sprint.

(A → B) To improve the communication with the stakeholder and to make sure that the team prioritises the most important features we could change our way of working in different ways. One way to improve this could be to have a meeting with the stakeholder where all members of the team are invited to make sure that everyone is receiving the same information from the stakeholder. Another option could be to coordinate questions from all team members so that our product owner could take all questions to the stakeholder in one meeting.
When we create user stories in the beginning of each sprint and divide our work between different team members we should be very careful to consider that no dependencies between user stories exist that will cause problems during the sprint. If we find out that is the case, we could try a different way of structuring our user stories for that sprint.

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) One of our goals was to improve in the working environment android application development. Some of the team members had previous experience of Android Studios, while others had never used it earlier. Thus the team had different starting points. The team has come a long way, however the improvement depended mostly on what each member wanted. Some improved in designing and others improved in other fields like api or database.

Another thing the team wanted to achieve within the project was to learn how to use the Scrum methodology. Scrum was unknown to the majority of the team at the beginning of the course. One team member had some knowledge about it and could in the beginning help with explaining concepts. Within Scrum we learned how to use a Scrum-board, sprint planning, daily Scrums, sprint review and sprint retrospectives.

Lastly we wanted to achieve a more customer focused mindset compared to the more traditional "code for yourself" style of other courses. In the beginning of the project if obstacles showed up and we needed an alternative solution we didn't ask the stakeholder, which impacted our sprint reviews early on. As the project progressed every change or feature was discussed with the stakeholder.

(B) In a future project the team would be even more successful if the success criteria had a better follow up. We focused more on our KPI:s than our success criteria because they were not as abstract. In a future project we would have more specified success criteria as a team and as individuals. The success criteria would change depending on which development environment the team would choose but achieving knowledge about how to use Scrum would most likely still persist. Another success criteria for a future project could be our teamwork and how the team evolves.

(A→B) To get more specified success criteria the team would have to be more detailed and specific about what we as a team want to achieve or as individuals. We could make it as much of a priority as our KPI:s at every retrospective and reflect on how a particular sprint helped in achieving or not achieving those success criterias.

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) We had a fairly standard pattern for our user stories throughout the course, however, some later written user stories started deviating from this, for example by not having a concrete description of the task at hand, more just a describing title. This was due to us writing most of the user stories for the entire project in the first week, and later realising that we were missing something. At that point it was very clear what we were missing so we felt like a more detailed description wasn't needed. This didn't have any noticeable negative consequences for the team, but we have seen that there was a risk that this low level of specification could affect the project

negatively and we would definitely make sure to follow a standard pattern diligently if we were to redo the project.

Task breakdown was something we didn't use in the beginning, likely because at the time of creating user stories we realised that in order to do a proper task breakdown, one would have to think deeper into exactly how the specific user story would be implemented. We did not feel like there was enough time to do this properly for every user story at the time. Later in the course we started doing a task breakdown for some user stories, which had a positive effect on our work flow as it became much more clear how far along work had come on a specific user story. Specifically for ones with higher velocity it was very useful to see what subtasks needed to be completed and such. We believe this task breakdown improved our productivity and in extension we created more value for our stakeholder by getting more work done in a similar timeframe.

The effort estimations did not come to affect our process in any major way. In the early stages we were so unfamiliar with the development environment that much of the effort behind a user story did not come from the user story itself but rather from having to familiarise ourselves with our set of tools. Since weeks also varied greatly in how much time people could dedicate, stemming from other courses, holidays and such. There was not really a steady level of effort reached. Effort estimations could likely be more helpful in a longer and steadier project than this.

User stories were significantly more structured in the beginning of the project, specifically the *definitions of done* grew to be quite vague and implied rather than specific and specified. This did not cause any major problems, as by the later stages of the project the group was so familiar with what needed to be done that there was not any major confusion. The group members also specialised in different aspects, if there had been more sharing of responsibilities, the need for more detailed user stories and definitions of done would likely be greater.

(B) There are a couple of major points that we want to change if we were to work on a similar project again. First of all, we shouldn't have tried to create all our user stories in the first week, and instead work more incrementally. The idea behind creating all the stories in the beginning was that we knew what our stakeholder wanted the final app to contain and how it should look, but in practice that didn't work too well. Instead of this we would want to create user stories about a week in advance or similar to stay flexible and reduce confusion, as it was easy to miss a

user story that we needed to complete when our list was about 60 user stories. In order to keep track of a more long-term goal with the app we would have created a couple of epics that would vaguely define how the app should work and look, and later break that down to user stories.

Another thing we would do differently is to work more with effort estimation, which is easier said than done considering it's very hard to estimate how much time a user story would take before starting working on it, especially when it's our first time working with an agile workflow. If we worked more actively with effort estimation we could have used that as a KPI, which could have been more accurate of our work performed than our chosen KPI's for this project. Finally, we would also have liked to specify a definition of done that we could use for all our user stories throughout the project. We believe that would have improved the quality of our code and created a clear standard for the work we performed.

(A→B) We believe most of our problems in this regard would be solved by having more experience and knowledge of the agile workflow. For example, if we started the project today, we would not have defined all our user stories for the entire project at the start. More specifically, in order to achieve what we previously stated we should spend more time on defining every user story properly to a template with descriptions, task breakdowns, and sticking to that template throughout the project. We would define a better definition of done which could include criterias such as; unit tests have been passed, code has been reviewed and that the product owner accepts the user story as done. Finally, we should spend more time on working actively with defining the velocity of stories. This would be done by reviewing how accurate the velocity of completed user stories at the sprint retrospective and after that adjusting our upcoming user stories' velocity in regards to the previous weeks.

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

(A) The acceptance tests have mostly been performed by trying out the app and the specific features, trying different scenarios to make sure that it's very unlikely that a bug will occur. This was done by the developers themselves, as they know best how the features are supposed to work. On top of that, the stakeholder tests the app every week which gives a different

perspective, focusing on if things are working in an intuitive way. We believe that with these two perspectives we have great coverage of preventing bugs and making the application run as it is expected to, and this is the resulting value of our acceptance tests.

(B) However we would like to add more testing within the program which will make it more secure that the application works as it is supposed to. So adding junit tests would make us more secure that we are able to build on our projects without worrying about a sequence of bugs which will result in more work.

(A→B) The reason why we did not implement junit tests from the beginning is that it takes a lot of time and due to very short sprints the amount of time that can be spent on testing is not that much, considering having only 8 sprints with a schedule that is kind of busy. However, we could make the sprints longer so that we have more time. Or we could take on less user stories in each sprint. Either way we would need to prioritise the testing more and possibly make a specific person responsible for the testing just to make sure that it is done sufficiently during the whole project.

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

(A) We have used all three of our KPI's throughout the project, and they are as follows:

1) How satisfied is the stakeholder

2) Percentage of finished commitments

3) How many stories were finished

The first one was good for keeping track of our customer satisfaction, allowing us to evaluate how the project is coming along in regard to customer value. The second one was used to evaluate if the workload we took on for the week was reasonable or if it was too high/low, this one proved useful for us and we used it frequently to adapt how much work we took on. However it can be misleading as often, the stories that weren't finished on time just had a bug or two that prevented it from being considered as done, but is represented as though it had not been

worked on. The final one served the purpose of tracking how much absolute progress we were making in our project, but we have come to realise that it is not too accurate as the effort estimation of stories differs a lot.

(B) For a future project we are satisfied with the first two KPI's as we have found them valuable in quantifying our progress in different perspectives. The third one we would have replaced with one that takes velocity into account, this would be more accurate and therefore more valuable as an analytics tool for the group.

(A→B) In order to introduce this new KPI, we would have to put more work into actively working with effort estimations at every sprint retrospective and sprint startup, as it would not be accurate and therefore not helpful for us if our effort estimations aren't at least fairly accurate. If we were to do this we could track our absolute progress more accurately and use this to evaluate our sprints more effectively.

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.

(A) We had a well defined social contract which we created in the beginning of the project. Since the work in the team faced no major problems related to the contract we did not have to go back to discuss its content much during the project. During the project we did some minor changes in the rules defined in the contract. We stopped to take minutes during our meetings after some time even though we had decided to do so in our contract. Even though we were aware of the change we did not go back and redefine this rule in our contract. We did not feel like this change in our process during meetings had a huge impact on our work. It is however as a general rule nice to have written documentation of meetings to be able to go back and check on what decisions were made there.

Regarding our teamwork, there were some problems in the beginning to set meetings for the whole group, but the team talked about this issue and managed to solve it.

The members of the team have chosen to work in different ways during the project. Some members preferred to work on their tasks individually while others met up and worked together. The members that chose to work together thought that it was a good way of staying motivated as well as being able to help each other out when bugs and problems appeared. This was a good way to enable learning in the team. Pair programming was also used by these members. Ida could not run android studio on her laptop so she was always doing pair programming together with some of the members of the team.

(B) In a future project it would have been better to have an updated social contract throughout the whole project. As we advanced through the course some things were added to the contract so that everybody will follow. So the team should spend more time on figuring out points that may come up like meetings and the quality of the work.

(A→B) If there were any changes made in the contract during the project, we would then have to go back to the contract and write down the changes. This would be a great way to make sure that any changes made in the contract were done consciously and not just happened to change because the team had forgotten some of the predefined rules.

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) On average the team has spent around 15-20 hours per week per person. In some sprints when team members have been absent there has been a lower sum of hours spent on the course in total. Usually this has been compensated by putting in more hours on other sprints when a lot of bugs appeared and when we took on a lot of user stories.

The work delivered during each sprint has differed over the different sprints. Some weeks our delivered work has been less than others, not because less time was spent, but rather that more problems were encountered which made certain user stories take longer than expected. The progress made measured in finished user stories did not always go hand in hand with the time spent.

The numbers of work hours did differ quite a lot during different weeks since there were quite a lot of holidays during the spring. This made the workflow during the project feel a bit disrupted.

(B) In a future project it would be nice to have a more even distribution of work hours over the sprints. The team could take on a bigger responsibility to help each member to maintain a reasonable and even workload over time. This could help every member to not take on too much or too little work, and to distribute the work evenly over every sprint.

(A→B) This could be done by using a shared timetable instead of individual, and use these for discussions in the team regarding how the workload was distributed between members, over the sprints and over the project as a whole.

Design decisions and product structure

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

(A) We have tried to use patterns that would make our project modular which will make it easier for continuing upgrading the project if the stakeholder decides to do so. A pattern we used in the project is MVVM (Model View ViewModel) which in android studio is a pattern used to organise the project and make it possible to build upon the project as well as making it easier to follow for new programers.

The API that we chose was also great as to how much it offered without any limitation on calls. Moreover it is for free.

We have also used singletons to access a single object and also tried to make as many interfaces as possible. Something that came up is also calling the API which was done on the main thread which is not good because if the api call fails the application fails. So what we did was to implement it in a parallel way so that the main thread still runs if the API fails and also make it faster.

(B) Things that could be much better is to get an idea of how patterns work before starting to work on it. For example, working on MVVM was not that smooth as some of us were following the implementation while others did not know how to truly follow it. However the api implementation is done with this pattern. What could make it better is actually documenting the code directly after writing it as to how much help it is to understand the code.

The API was great however when searching for a specific movie we had no choice but to implement it on the main thread which is very bad. Because if the API call fails it can become a problem even though we have done it in a way that prevents the app from crashing.

(A→B) We should spend more time on planning and communicating on what should be implemented and how it should be done. And also we should be willing to take risks and learn about the best ways to implement things.

And for the API we should think and ask for better ways to implement multiple calls. We should also study the API before using it.

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 Scrum-board in Trello which was great to split tasks and describe them. In the beginning it was very hard to estimate and describe the tasks as having to deal with a new environment. We have also written some protocols in the beginning for our meetings and some documents for planning our work. Moreover we have done a UML-diagram of how the project is going to look like. However, due to being unfamiliar with the new patterns we are supposed to implement it was hard to visualise the diagram.

(B) We should continue using Trello, however we should spend more time on planning and also describing the tasks. We should be using more diagrams like UML and user cases which will make it easier for us to follow and implement new and current features. In future projects it could be beneficial to use and update an UML diagram throughout the weeks to better map out the code architecture. This could be beneficial for applying object oriented patterns and principles as

well. We should also continue documenting our meetings so that we have a source to look back on and check for important things that may have come up.

(A→B) To do these things we need to clearly state rules from the beginning in the social contract which makes documenting meetings as well as spending time on planning for the user stories possible. As to the diagram we should maybe take a look at how a project of this type may look like in this kind of environment and take inspiration before starting, and also continue updating the diagram.

How you use and update your documentation throughout the sprints.

(A) The most central documentation the team used was the Scrum board in Trello. It had all our backlog stories as well as the stories for the sprints. We had the columns “Backlog”, “ToDo”, “InProgress”, “Testing” and “Done”. Stories were moved and updated according to status within the sprint. The person or persons responsible for a story were the ones to move the story. If some stories were not in the right column by our retrospective, the team moved it to its proper column. Sometimes user stories were updated with extra tasks when problems showed up in the middle of a sprint. We also wrote down bugs related to the user stories, so that anyone in the team could take a look at the problem without having to ask what problems exist. As stories were updated, we were meticulous at pushing updated code to github whenever someone made changes.

We added comments and Javadoc in our code during the project. Not so much during the start, but more and more as the project moved on. Somewhere in the middle of the project the team decided that the Scrum-master should send out reminders to comment code, which the Scrum-master did. All team members had an expectation to write comments on their code.

In the beginning of the project we documented our meetings but as the project moved on we did so less and less. Sometimes we forgot what we decided at different meetings, so some documentation would be good.

(B) In a future project we would continue using a Scrum-board because we felt it gave us a lot of structure and was easy to use. We would also try to comment on code and use Javadoc from the start instead of applying it later in the project. Going forward it would also be beneficial to

document certain decisions at meetings. We didn't find documenting the whole meeting beneficial but bigger decisions regarding the project outcome, expectations and such would be good to write down in the future.

(A→B) To comment code earlier the Scrum-master could keep everyone accountable for writing them down and checking if there is code without comments/Javadoc at the end of the sprint. To make sure that important decisions are documented the team could assign a secretary role to someone during the meeting.

How you ensure code quality and enforce coding standards.

(A) Our group decided early on that we would follow certain naming conventions for classes, packages, XML-files, etc. :

- Capital letters for classes
- Lower case letter for packages
- Lower case letters for XML-files and underscore between words

Although these guidelines were put in place at the beginning of the course, sometimes it has not been applied. Therefore some variations exist. We did not set any big cover to check whether every member has done it the right way.

One particular part of the code that has absolutely no consistency is Id's for components in XML-files. They do not follow a standard and vary greatly. This is due to the fact that we never talked about the naming of these and also that some parts of the files were done by following youtube tutorials, where they had their own naming conventions.

(B) In a future project we would like to have written code which is according to a set guide of naming conventions. We would like to follow the guide from the start and throughout the project.

For future projects we would've made sure that we scheduled time in the beginning to write down a set of rules and guidelines for naming conventions and coding in general. By doing so the group would have a guide to follow from the start, which would decrease the need for refactoring later on in the project.

Having a strict set of rules for coding standards etc also makes it easier to troubleshoot if something goes wrong, since you won't be as confused by the great variation of code and names. We noticed this during our project and it is therefore another reason for why we would like to do it differently in the future.

(A→B) To implement the improvements above there are two things we would do. Firstly, as we already have mentioned, we would make sure that we schedule time, at the start of the project, to decide the coding standards and naming conventions we would like to implement for the project. Secondly we think it would be beneficial to have a dedicated person within the group that observes and makes sure that the group follows the set of rules that we have defined. The person would control the code, during and at the end of every sprint to ensure that we consistently follow our guidelines.

Application of Scrum

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

(A) Jenny was our product owner during the project. The product owner had an important role since she had weekly meetings with the stakeholder. The feedback from these meetings were translated to the rest of the team and were the main source for what the team should focus on to create customer value during the sprints.

A different scrum-master was chosen at the start of every sprint. We chose this setup since we wanted to let different members have the opportunity to try on this role. It was quite hard to take on this role for the first time, especially early in the project when the concept of scrum was very new to most of the members of the team. After some sprints it was easier to understand the working process and what part the different roles played in it. We realised that when the scrum master was spending more time communicating with the team and trying to connect the team with each other, the work was more effective and the team members felt less stressful.

(B) In a future project we would like the scrum-master's work to be consistent and focus on maintaining effective communication within the team.

(A→B) To be able to do this we would need to specify what kind of responsibility the scrum master has and what actions are expected. This would help the scrum-master to work efficiently in every sprint and to help the team moving forward in the project.

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) Learning new tools and technologies has worked pretty smoothly during our project. The experience of working with some of the tools varied between the members of the group. Only two people in our group had worked with android studio before but it wasn't too much of a problem. Having the experience of learning from other courses we knew that youtube and google are tools that would be very helpful for designing and also programming and familiarising the IDE. Youtube was great for solving problems or researching different code/design solutions. Google was useful for searching for common bugs and also some coding solutions.

Another tool that was new for some of our group members was Trello. However, learning to use Trello was easy and the learning journey was simplified by having experts in the team.

Another great way of learning has been to work in a group, especially when familiarising with the new coding environment. In the beginning the group members that took on the responsibility of developing the front end of the application, sat down together and learned how to use Android Studio. Later they started working more independently, after having some basic knowledge, and further development of skills could be done through online sources.

(B) Generally we believe we had pretty good ways of learning during our project but we do think that we could have improved it in two ways. Firstly we didn't really use the opportunity to get help from the teaching assistants or examinator of the course. So for future projects we would like to have more contact with them, as it probably could save us some time when having certain coding problems or just general guidance in how to work most effectively with the agile principles.

Secondly we would like to expand on our group learning method and maybe dedicate time to have actual workshops within the group. We think that would be both very enjoyable but also enable the whole group to learn from each other, even if it's not necessarily things that you will have responsibility for coding later on.

(A→B) In order to make "better use" of the TA's and examinator of the course we would make sure to have a dedicated list of questions prepared for every supervision session. That way we could gather all of our thoughts during the sprint from the week before and then brainstorm solutions with the TA or examinator.

Also we would schedule time in the beginning of the project to have some workshops within the group, so that we all can familiarise with the tools and technologies together.

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

(A) The user stories and how to work with these was something we focused a lot on in the beginning of the project. In the beginning we spent quite a lot of time on writing the stories. However, as the time went by, we became more and more confident in the project and our work with the app. This meant that we felt that we did not need to describe each user story in detail as we had in the beginning of the project. However, in some cases, this affected our way of working negatively since user stories with a detailed description felt more doable than the ones without.

Every week started with a sprint planning session where we created new user stories, added their velocities, and assigned them among the team members. This work was not very structured in the beginning, but after some sprints we felt more confident in how to do these meetings. To use the concept of velocity was something that we struggled with and thought was tricky even in the later stages of the project.

Because of different schedules, the group decided to have digital daily scrums via messenger. In these digital scrums we discussed the progress so far, answered each other's questions and sometimes gave each other to do tasks.

During the first two weeks, the stakeholder reviews were scheduled on Thursday evenings. Later, we decided to move the stakeholder reviews to Friday mornings instead. This gave us an opportunity to do some late fixes during Thursday evening before presenting the work of the sprint for the stakeholder.

Every week on Friday afternoon we had the sprint retrospective meeting. During the retrospective the product owner retold the stakeholder's current opinion about the application. We also discussed the progress in the sprint and the project as a whole, and discussed if any changes had to be made moving forward into the next sprint.

We were using task breakdown in our user stories, but this was mostly used at the end of the project.

We had a product backlog in our Scrum board. Unfortunately we did not spend much time in keeping it up to date, and the list of user stories were not ordered after prioritising level.

Overall we became better and better each sprint at applying SCRUM practices, but there is still a lot of room for improvement.

(B) The first thing we would do differently in a future project is to have daily scrums in person, as standing meetings, instead of quick digital check-ins. We believe this could facilitate better and more direct communication within the group. It would also help with ensuring that every group member is on track with their user stories and it would be a time to effectively address any potential issues. That way the group could work together to solve problems that might hinder the team's progress.

The second thing we would do in a future project is to improve our use of the backlog on our scrum board. During our project we have not been very strict on properly ranking the user stories in the backlog. By making sure that it is properly ranked it would give a better view on what the group needs to do next and what we should prioritise for ultimate customer value.

Lastly we would make sure to dedicate time to sit down and do the task breakdown. In that case we could either do them all together as a group or we sit together and do the task breakdown separately first and then the group goes through them together. By making the task breakdown a routine for every week, it wouldn't be overlooked and it would decrease the risk of missing any

details/tasks about the user story. Missing details about a user story could mean missing important requirements, which in that case loses customer value.

(A→B)

To implement the daily scrums in person, we would need to schedule this meeting at the beginning of the project. To improve our work with the backlog we could dedicate time for this during our meetings. It would be helpful to have a list of tasks/areas to discuss for each of our meetings where updating the backlog would be one task on the list, to make sure it is done regularly throughout the project. The improvement of the work with task breakdown could be done as described above.

The sprint review and how it relates to your scope and customer value.

(A) During our project we have had one sprint review per week (Fridays) and we think that has been very rewarding. During the sprint reviews we first took a look at our scrum board and evaluated how the sprint had gone, how much progress we had made and whether there were any tasks or user stories left on “to do” or “in progress”. We continued to discuss the potentially unfinished user stories of the week and also how the week had been in general. We then tried to decide how to deal with the tasks that were left and whether we needed to change anything for the next sprint.

During the meeting we also went through the feedback we got from our stakeholder. Jenny (the product owner) had meetings on Friday morning with the stakeholder to get a review of our progress. Often we got some notes from those meetings, which we then had to decide on how we should deal with. Sometimes there was something small the stakeholder was unsatisfied with and sometimes it was more extensive, which affected the way we would work with the feedback. But regardless of the extent of the feedback we made sure to take it into account for the development ahead.

The reviews we did not discuss the following sprint or any new user stories unless it tied into the feedback from our stakeholder. In that case we either made a new user story, if there was some entirely new information, or we reevaluated an existing user story and potentially added new tasks. We always wanted to make sure that our vertical slices were finished before we moved on.

Therefore we prioritised the feedback highly and wanted to address any potential problems or dissatisfaction with the product before we moved on to the next user story. In that way our customer felt satisfied with the delivered parts of the application every week.

(B) Depending on the relationship with the stakeholder, it might be beneficial to include them further in the sprint review. This would of course depend on the availability of the stakeholder as well but if the stakeholder could be present during the meeting where we discuss the feedback, it would enable a more direct and extensive information channel. It could also be interesting to have the stakeholder present in the planning phase for the next sprint since in that case it would be our customer that decides what to do next and not us. That would maximise the customer value as we might not have the same way of prioritising as our customer has.

It could also be interesting to have more of a hands on communication with the stakeholder when defining the Dod of the user stories. By doing so it might be even more clear for the group where the priorities are and what needs to be done.

(A→B) To create an even better communication between the stakeholder and the team in a future project we would schedule some appointments with the stakeholder, where the whole group could communicate and present the progress of the product.

Alternatively, if the stakeholder is unable to participate at our meetings, we would prepare questions for the product owner to ask during the demo before the group meeting.

Relation to literature and guest lectures.

(A) The main sources of inspiration for Scrum and Agile principles were the lecture slides. The lecture slides were most useful, in the start of course, approximately during the first three weeks, when the team members were learning what Agile and Scrum means in practice. The Mona Lisa exercises session was a useful experience in a more practical way. The Mona Lisa taught us the basics of sprints. After the spring break, our team felt more comfortable which meant that the main focus was shifted to practical parts, like Android app development, and finding relevant tutorials and Stackoverflow pages to get the things done.

(B) The best approach would be to also seek inspiration from other sources who have excelled in agile, whether it be teachers, other professionals, or literature. However, as we were taught in lectures and supervisions, there is no one-size-fits-all approach, and each team has its own dynamic and needs to discover its own form of agile work.

(A→B) Next time we are doing a project, we should revise the material a little bit each week, and look more for other sources of agile and scrum examples in real life.