Kashyyyk Final Report

Authors: Wincent Stålbert Holm, Isak Holmdahl, Viktoria Hagenbo, Lovisa Rosin

Oscar Marrero Engström, Rasmus Standar, and Marcus Phu

Customer Value and Scope

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

Our initial initial scope was defined at the start of the project while writing our business model canvas. This was done by deciding which target group the application should be aimed at. In our case that would be people who would like to stay active by walking or running, and that want to be able to explore new areas while doing so. From the perspective of the target group, discussions were held to see what features would be interesting to have in our application, and from this a mockup was created. After designing our mockup we decided on our MVP (Minimal Viable Product), which basically included the route generator and a user interface making it possible to navigate the application. We still chose to have features not included in the MVP in our initial scope, but with lower priority. Although we had some prioritization, the priority for each individual feature was not clearly defined. Instead, this was something that was done on a weekly basis by our PO (Product Owner) during the planning phase of a sprint.

After a couple of sprints, when reflecting on how long we had come, we decided to narrow our scope down to finishing only our MVP. This was to ensure that we had enough time to finish our core application.

For the next project, we think it would be better to set up a narrower MVP and have this as the initial scope. Then if the team feels like there is time later on, more features can be added to expand the scope. To make the scope clearer and to help the team work more efficiently towards the scope it would be ideal for the group to discuss and write down the priorities for each feature of the scope at the start of the project. When deciding on the priority it is important to think about the stakeholders connected to the user-story or features, as it is important that the features that benefit the key-stakeholders or target group should have higher priority. Additionally, it is also beneficial to make a time estimation of each feature in the planning phase of the project to help the group decide on a reasonable size for the scope.

Based on the knowledge received from this project, we know that the next time we will start a similar project it is better to start off with a smaller scope and eventually expand the scope later on, rather than narrowing the scope during the project. To make sure that the scope is

reasonable it is a good idea to write down all user and developer stories and decide what priority they should have. The priority here should be based on how much value it brings to the stakeholders and the end-user but how much time the team thinks will be needed to complete it should also be factored into the priority. If a task might give plenty of value but will take up all the time available, it may be better to prioritize a larger number of smaller features that will sum up to a greater value in the end, and that can be achieved. When thinking about the time a feature or story will take it is important to try to estimate also how much time the developer needs to put into learning new skills, which is something we as a group now feel more comfortable doing.

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)

We did not set any clear success criteria regarding the final product, but of course we had a MVP and this was seen by the group as the success criteria for what we would see as a successfully developed product. We did set some success criterias that only spanned one sprint and these were focused on what we would like to learn during these specific sprints. One sprint the success criteria was to learn Bootstrap for example. A success criteria that again, was not clearly defined but rather spoken about when starting this project was to learn more about web development. There was also a success criteria that the group mutually agreed upon verbally when starting the project and that was to not stir up any conflicts that would negatively affect the workflow, and the group made sure to successfully achieve this with the help of the social contract. Conflicts can be good to spark new ideas, but when they negatively affect the amount of work done and the group's mood then these conflicts only become a burden.

Some success criterias existed regarding what the group wanted out of the application, learning experience and teamwork, but they were not formally defined. The group can not be sure that everyone in the group agrees to these success criteria or even knows about them. A discussion regarding the specific success criterias for these topics should be held. By having such a discussion at the beginning of a project the group can be sure that all group members understand the criteria and have had a chance to be heard and affect the criteria. By also reflecting on these success criterias every sprint, or even every workday, the group can be sure that progress is made and that any changes that are relevant can be made to the criteria. These changes could be necessary if the circumstances somehow would change. Then the criteria set in the beginning might not be realistic anymore.

To make sure that criteria for teamwork and effort given are met, a social contract like the one made for this project should be maintained. The contract worked well and did not have to be altered too much after its first version since the group made sure to take the necessary time from the beginning to ensure that everyone agreed and that all bases were covered. The process for creating and maintaining a social contract would therefore not look much different in future projects.

One last note is that the team found that success criteria for learning outcomes should not be forgotten, since a success criterion can be to learn enough regarding the tech-stack inorder to become a cross-functional team, something that greatly helps current and future projects with similar tech-stacks. To achieve this the team should early on research what technologies will be used in order to have sufficient time for every member to learn about these chosen technologies.

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

Looking back at our user-stories, a clear development can be seen through the project. In the first sprint our user-stories were often formulated to follow the criteria of INVEST but they were lacking two key aspects. The first of these aspects being that we had no clear acceptance criteria, which worked out all right in the beginning as we were working on user-stories where the acceptance criteria could be understood without needing to be formulated. These stories were often to create a new empty html-page and make sure that there was a button that could be pressed that led the user to the empty page. But as we started working on more complex tasks the need for acceptance criteria was noted, and we started to include it into the stories to help the developer to know what needed to be done for a user-story to be completed.

One more aspect that we added to the user-stories after a while was the estimated effort to deliver it. This effort was calculated by the usage of planning poker, where all team members first got to assess how much effort they thought would be required. This was followed by a discussion of why the persons that had highest and lowest estimated effort thought like they did. After the discussion everyone got to vote again, and this result was used as the estimated effort for the user-story. By doing this, all the team members got a better understanding of what stories the team had, but it also helped the PO to make better plans for what to be prioritized during a sprint.

After having learned the perks of writing good user-stories, the team agreed that the next time we do a project it is important to take time, especially in the beginning, to write good user-stories. Here good user-stories are stories that follow all the criteria in INVEST. Not only do we think it is important to do this, but we also think that some sort of acceptance criteria need to be formulated and that a first estimation of effort should be done at the beginning of the project. Here it is important to think about all the types of effort that can arise, for example a user-story might require more knowledge and then the process of obtaining said knowledge needs to be taken into consideration, not only the effort for completing the user-story once knowledge is in place.

Additionally, we think it might be a good idea to think about what stakeholders are connected to a user-story and clearly define this in the user-story. This is to make sure that the developer understands exactly who the target is for said user-story.

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

During the planning phase of the project there was not much effort put into thinking about how our acceptance tests were supposed to be conducted. In our definition of done we mention writing tests (unit tests) to verify that the code works as intended, but during the project very few unit tests were developed, and the ones developed were only used by the developer and never during the presentation for the PO. Another mistake we made during the project was that we did not plan for creating unit tests or clarify how the acceptance tests were supposed to be conducted during our Sprint planning sessions. This resulted in our acceptance tests being done by creating a pull request and tagging each affected team member to review and doing some black-box testing on the code before merging the branches into the main application. The presentation to the PO was also one kind of acceptance test via black-box testing since we gathered feedback from the PO which we then applied for an even better solution. This way of testing worked fine during the first weeks when the front-end of the application was the priority and it was easy to identify the problems and bugs but when we got to the back-end it was harder to conduct the black-box testing. For example, in one sprint we had developed a system to generate a route that the developer thought was acceptable but after the code review and black-box testing we decided that the system was too flawed and not delivering a route within the margin of acceptance and therefore we scrapped that system and developed another one for the week after. Thus the week after this situation occurred we were more thorough and aggressive with our black-box testing, asking for feedback for each iteration instead of the final iteration.

Looking back at it, had we properly defined how the system should have been working and what we expected we probably would have caught the flaw earlier and maybe could avoid having to scrap the developer's weekly effort.

Though in the end these black-box testing sessions generated tons of value for both us and the stakeholders. During the black-box testing and code review we learned about how the application worked and in some cases we even learned new ways to approach different programming obstacles from each other. For the stakeholders this generated value in the form that the application had a higher quality for example if these tests were not conducted and we kept the flawed system for the route generation, the application would have been worse and therefore be of lower value for the end users.

For future projects we all agree that it would be favorable if each task had clear and concise expectations defined before starting the development of each task. To achieve this one option would be to have a discussion about what is expected, how it should work and what’s the MVP for this function. This is to clear up possible misunderstandings and to make sure that everyone has the same acceptance criteria for the task. Another option would be to already at the beginning of the project, clarify that unit tests have to be done and have the PO require it and wants to see it before considering the task to be done.

Another subject that we think would be helpful for future projects is including and planning for the testing and time to develop the unit test into the sprints. To achieve this it would require the Scrum Master to take testing and unit test development into consideration during the sprint planning.

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

The first KPI we used was Velocity which we used by having an effort estimation done through poker planning. By doing the effort estimation we got a value of how much effort we thought each task would require. By knowing the value of each task we could deduce a velocity for each week that we could compare to previous weeks and then adjust the workload of each week by either reducing or increasing the velocity. This was very useful during the planning phase since we could prioritize which task that we must do and which task that could be in the backlog, but also adjust the stress level of the team. One example of us failing to do so was during sprint week 6 when most of the members had a lot of other obligations in their other courses. Even though this was announced during the planning phase we set a velocity almost as high as the previous weeks. For this week we did not succeed with all of the tasks we set out to do and hence the team morale went down. But other than this particular situation we as a group feel that velocity was a great KPI to use and that it was a very beneficial tool for planning each sprint week.

The second KPI we used was Cumulative Flow which we used by marking how many tasks were done, in development, in need of testing or moved to backlog. The intention of this KPI was to monitor if there was a hiccup and somebody needed help or was stuck with a task since it would then show as a task in progress at the end of the week and the Scrum Master could then redistribute the effort in the group. The problem with this KPI was that it never could be used as intended since most of the work was done on Thursdays and Fridays when we all could sit together as a group. This led to the pie chart not moving at all for 3 out of 5 days and then going from red to green instantly. So in the end it was never used by us as a group and did not benefit us.

As a group we think that this would probably be used better if we had a designated Scrum Master that could focus and catch these hiccups - since during this project the Scrum Master was also flooded with developer work - and a more realistic workflow since then it would mean that work had to be done during the whole week instead of just two days of the week.

The third KPI we used was the amount of tasks done and represented by a burnup chart. This was used to monitor our progress of the week but kind of fell in the same trap as the Cumulative Flow KPI did as it only showed progress in the last two days of the week.

The overall feeling in the group was that KPI:s were useless outside of the Velocity KPI since they did not update until the last two days which we then already knew would solve itself since we worked together on campus. Another reason we think why we feel that they were not as useful is that we were late at starting to use them thus making them feel a little unnecessary.

For future projects we have to be better at identifying which KPI:s are useful for the specific situation and when it is suitable to use a KPI since they have shown that they could generate value for us as a developer, for example improving the planning and adjusting the workload. Another thing we want to do for future projects is to integrate KPI:s into the work process as it would keep the developers updated about the progress of the project, for example if we had daily scrum meetings and the Scrum Master shows the KPI charts every day we would be able to continuously use the KPI as intended.

Based on the knowledge received from this project, the next time we would identify KPI:s that are more appropriate for our situation because in this project the only appropriate KPI that we chose was Velocity since both Cumulative Flow and the burn up chart with tasks done was not used due to them not providing any useful information until the last days when we were working together anyways. Instead we should identify KPI:s that have more meaning for the project, especially now that we have experience working with KPI:s and some knowledge over how it is supposed to be used. Here our expanded experience working with KPI:s will matter a lot for future projects since it is hard to pinpoint each pro and con with each KPI without prior knowledge.

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)

During the first week of the project we wrote a social contract containing rules, practical as well as social, that every team member was responsible to adhere to. This contract remained static throughout the project, apart from some minor additions early on. That is not to say we did not revisit it from time to time, but that we thought it served its purpose well the way it was formulated from the start.

One purpose of the social contract was to have some guidelines in case there were any conflicts or disagreements within the group. During the course of the project we did not have any major conflicts, which is why we did not get to use the social contract in order to resolve any conflicts. However, by taking time to formulate a social contract and making sure that every team member gets to disclose their expectations and limitations can in and of itself prevent misunderstandings and frustrations. From the get go, the entire group had a clear understanding and agreement of the ambition of the project, which we think can be a result of having a social contract. In that sense, our social contract might have been a useful tool for conflict resolution, albeit not in an active sense.

In our social contract we stated that every meeting was to start with every team member getting some time to inform the rest of the team what their expectations, worries, and overall feeling regarding the project was. This proved valuable to us. Especially since we found that

many, if not all of us often shared the same thoughts and feelings. This made it possible for us to early on find the bigger issues and adjust our work accordingly.

For future projects we plan to keep the “round table”- discussion, as well making sure to spend time on making a thorough and well formulated social contract that everybody agrees on and understands. As stated above, we were throughout the project satisfied with our social contract. However, we do not exclude the possibility to make it even more detailed and “waterproof” in the future. We also recognize how formulating a social contract is a good opportunity to get to know the other team members, which is valuable when working together.

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)

Overall the team’s average time spent on the course is 15-17 hours per person and week. Most of this time was allocated into group meetings that evolved into programming sessions, both on campus and over zoom. Throughout the project, a sprint-planning session was allocated on mondays and usually totaled to around four hours. Each week we also set time for sprint reviews at the end of the week, during which we reviewed the progress of that week. If we had time, we followed the sprint review with preparing for the planning-phase of the upcoming week. This made the planning easier, more efficient and of higher quality.

We believe future projects would benefit from better time management and making sure that the time spent working is used as efficiently as possible. Higher quality of the hours spent and less overall time spent should be strived after. We found that some weeks, especially those when our planning had been lacking, the efficiency of the group overall dropped. For instance, there was a clear correlation between the velocity of the group and the quality of the user stories we were working on for the week. In order to prevent this from happening in the future, more time should be spent on planning and making sure that our user stories meet the INVEST-criterias. We think this could reduce the amount of unnecessary work being done as well as less time spent on being unsure of what the next step is.

We also think the person acting scrum master should include the time spent on preparing for the upcoming sprint into the calculations of the week’s velocity. This would make it easier to ensure a good workflow for the coming week.

Design decisions and product structure

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

During this project a web application with an extensive backend was developed. The application’s frontend is built with the help of primarily Bootstrap and Thymeleaf. All of these technologies together with the basic techstack behind a website was new to most group members, but something everyone felt comfortable learning. Our choice of Bootstrap as a

stylesheet library assisted tremendously with giving the website a modern and polished look; and Thymeleaf gave us the possibility of localizing the application to different languages.

The backend development was built with the Java web framework Spring Boot. Being a framework it locked us into a standardized development architecture and set of design patterns. It gave us access to a lot of utilities which assisted with development, and in the end a better end-user experience.

Our focus in the backend was mainly around handling map data. During the initial planning phase we speculated on what data sources we might utilize. For our use case we required quite a lot of data, ranging from where walkable paths are located, the elevation throughout those paths, and their locale. We ended up putting a lot of resources towards using the Swedish “Lantmäteriet”, as it promised to provide all of the aforementioned information. That high resolution map data we thought could contribute to the value of the product – but it turned out getting that data was quite difficult. What we had neglected to consider during the initial choice of API was that not every provider gives out their service to everyone. Lantmäteriet required the signature of a contract and only provided bulk data, both qualities which made it unsuited.

What we ended up using was a combination of OpenStreetMap and OpenElevation, both completely free and open source solutions. Their data is sometimes of lesser quality when it comes to Sweden, but with the benefit of providing data for all of Earth. This we believe contributes significantly to the value of the product, as we can extend our service to cover every part of the planet, not just Sweden.

The group believes that the choice of techstack worked well, and everyone is happy having learned a couple new technologies and concepts. Nevertheless the development process could have been streamlined with more time spent with the initial planning. If we had spotted the difficulties with getting map data from Lantmäteriet we might have had more time to develop our solution with OpenStreetMap. This is something we will take with us into future projects.

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

Technical documentation was not systematic in this project. Most of our technical planning happened sporadically using a whiteboard and pen. This was useful to sketch out the next steps, but not proposed to serve as a looking back point. During most sprints we were never all in one place at the same time, which made it difficult for everyone to follow if something was done on a whiteboard.

Some planning was done with the use of online drawing tools. This made it both easier for everyone to follow and contribute, but also to look back at. Our mockup served as a looking back point during a large part of the development process, which was helpful. The planning of the conceptually difficult routing algorithm and its integration with the frontend we had more thorough technical documentation – this was also later into the project at which point

we had learned the value of having such documentation. Such documentation took the shape of sequence– and class diagrams. This gave a clear overview of what had to be done and how, which also yielded better task delegation.

A recurring point when it comes to the technical part of this project is that more time should have been spent at the start sketching up the application's integration. If we had utilized more technical documentation from the start it would have been easier to implement and delegate each part of the application. We might have also realized how difficult of a problem the routing algorithm would have become if we had done more planning regarding that – a very integral part of our application and its value – at the start.

How you use and update your documentation throughout the sprints

In the same manner as creating documentation – updating up documentation was not systematic. Code documentation in the form of in-code comments was the primary source of documentation when it came to communication to other group members how a specific piece of the application worked, and this worked well. If something substantial changed in the code that a member modified, the comments regarding it were regularly updated. What was seldom updated was drawings like class diagrams made to plan out a bigger portion of the application.

For future projects we would like to be more systematic with keeping documentation up-to-date. This could contribute to keeping everyone on the same page when major things change in the workings of the application. It would also be valuable for a possible product owner and/or client to have an overview of the application – if they were interested.

Introducing guidelines at the start of a future project for how creation and maintenance of documentation will work could be helpful. It could be added as an addition to the definition of done that any modifications have been sufficiently updated to reflect changes.

How you ensure code quality and enforce coding standards

The DoD defined a couple of rules for how code should be written in order to enforce a coding standard to make the project get a uniform code structure and to make sure quality did not drop as the project grew. The responsibility to follow DoD was put on the developer that was developing the task, but the code was also sometimes checked if another group member was working with a closely related task, and the code was also checked in the pull requests that were done in order to merge to the main branch. The quality and standards defined in the DoD could also be reviewed while group members exercised the peer programming technique.

The team is happy with how the application turned out, but a consensus was reached that DoD was not very strictly applied, but more hovered around in the back of one's mind and therefore might not have been enforced to the length that it should. In future projects these requirements on code quality and standards should be enforced stronger and checked more regularly, while not disturbing the workflow. The group also feels that a strict git protocol

should be enforced, since there actually was an incident regarding git early in the project that halted development for a few days. By creating a strict git protocol and making sure that the team understands it these types of incidents can be avoided.

By also creating more and smaller git pull requests the team can ensure better code quality, since the work of proofreading code won't be as big of a challenge. Also, if everyone makes sure to really understand the DoD for code structure then everyone would not need to review every pull request, instead only a few could do that. This also requires that the team has worked a lot on being cross-functional.

Application of Scrum

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

During this project, we have had the team members practice different agile roles. From the first week of the course, we decided to have one team member act as the Scrum Master for each week, and the rest of the group should act as regular team members. The Scrum Master's role was mainly to act as the "project leader" for the week, where their tasks would include putting user stories into the backlog, updating the backlog, and to generate the KPIs. We also agreed on having a round table at the beginning of each meeting, and the Scrum Master would be in charge of conducting them.

After a meeting with our supervisor, we were encouraged to have one of the members act as PO each week. The PO's assignments included prioritizing what user stories we should work on during the week's sprint, and also adding them to the backlog. They would also check if the DoD was checked for the user stories before asserting them as done, and also act as a judge when new functions were added. We presented our progress for the PO at the last meeting for each sprint, and the PO would then give their feedback regarding if they were pleased with the outcome.

The role of the PO was a bit diffuse, as the PO was also very much working on the project at the same time, which made the role blend into the regular team member role. We agreed that for the role of a PO to become more distinguished, they should not be a part of the development team and instead behave like an outside customer we delivered customer value to. This was hard for us during this project, as we also needed the member acting as a PO to work on the project at the same time.

The role of the Scrum Master did feel like it gave some value and structure to the group, as the Scrum Master was also a part of the developing process. It was a little bit diffuse during the first sprints, but later on we started to feel that having a person that was more in charge of the structure and process of the group was valuable. For future agile practices, the assignments for the PO and the Scrum Master should be more distinguished and they should preferably not be part of the regular developer members.

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

Besides having a Scrum master and a PO, for this project we have tried a lot of new practices that have helped us with our structure and progress while also being compatible with the agile approach. For instance we used planning poker for estimating how much effort each user story would take, which made it easier for the PO to assign tasks to the backlog. It also raised the tasks to discussion, which ensured that everyone was on board with what they meant.

Pair programming was also used a lot between different members since some of us had more experience in various technological areas than others. This was very much appreciated since it resulted in new insights and new skills for some members in these fields, and made it easier to avoid getting stuck with a task. The sprint-structure, with a planning phase at the start and a review at the end, was a way for us to arrange and distribute tasks for each member, and discuss what went well and what we should try to make better the following sprint.

We are all in agreement that the agile approach was a solid and productive way of organizing a project like this, and is something we definitely will consider for future projects. With a good foundation of what the agile way of working is and how it should be carried out, it leaves us with an opportunity to refine what we have learned so far, and makes it easier to learn even more as well as applying it to future projects.

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-prioritization of user stories? How did the reviews relate to your DoD? Did the feedback change your way of working?)

During the project we had a rolling schedule for the role of PO, making sure that everyone got to be PO for one sprint each. At the end of each sprint we allocated time for sprint reviews, during which the PO got to evaluate the progress made by the group and whether the work made had resulted in value for the customer. User stories were typically not prioritized during the sprint review, but during the sprint planning. However, the PO for the week could use the lessons learned from last week’s sprint review when prioritizing the user stories for the coming sprint.

During the review, we usually spent time reviewing pull requests and merging the branches with new code into the main-branch after they had passed the review. We did not take too much notice of our Definition of Done when doing this, especially during the weeks when most of the work had been done on the frontend. However, many of the points in our definition of done were always in the back of our minds when reviewing code, but this is definitely an area of improvement for us in future projects to ensure good code quality and uniformity. For upcoming projects, it could also be of benefit to spend some more time

formulating the DoD, making it a little less arbitrary and thus making it easier to verify whether the new additions meet all the necessary criterias.

For this project we have been working in a simulated Scrum-team with a PO that changes every sprint, which can make the prioritization harder than when this is not the case. We believe having the same PO for the duration of the project would make it easier to keep a red thread throughout the project as well as making it easier to formulate a definition of done that ensures that the product meets the expectations of the customer.

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)

After deciding on the scope for this project, it was clear that several members of the team would need to acquire new skills in order to face the challenges the development of it would pose. The majority of the team had little experience in developing a full stack application like this one and lacked the technical knowledge that would be required, especially for the frontend development. As a result, acquiring new knowledge in using certain tools and learning what would eventually become the building blocks of our project was a crucial part of the first sprints. This was reflected in some of the tasks on the scrum board during this timeline, which consisted in becoming familiar with and learning how to use frameworks like Bootstrap and languages like JavaScript.

The process of acquiring this knowledge was both an individual and a team effort. All team members spent a considerable amount of time reading up on material related to these topics. In addition, a smaller subset of the team did have some previous experience in frontend development and was able to share this knowledge with the rest of the team. Both the individual effort and the sharing of knowledge between team members were key factors contributing to the success of the development process.

Other aspects that allowed the team to progress steadily in the development of the application were practices like pair-programming, which consisted of two or more team members programming together. This ensured that the members participating in this activity held each other accountable for the functionality of their contribution while at the same time providing a safe space for learning since the more experienced participant could explain unclear aspects of the contribution to other participants should the need for it arise. Another tool that facilitated the development was the use of version control in the form of GIT. This was a tool that all team members were familiar with from previous experiences and it was clear to all that the use of it contributed to the structure of the application being developed. In addition, it also facilitated certain aspects of the development such as gaining an overview of what each member was working on during a given sprint.

For future projects the general consensus is that it would be better to minimize the spread of knowledge obtained while learning about new topics. Another possible improvement moving forward would be to incorporate the key practices that supported our success in this project and possibly adding to them to improve their impact.

Minimizing the spread of knowledge could be achieved by limiting the resources each member is studying to acquire new knowledge, perhaps by having a common set of such resources. This would contribute to all members having a more similar common ground to work from. Identifying the key practices that contributed to a successful development is also desirable, and this could be done by evaluating the main practices used in the development process. Improving on these practices would be an experimental exercise left to members participating in a future project in which a particular practice is wished to be worked upon.

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

During the course there was a guest lecture by Matthias Becker on agile practices and how this process is used in a real-world scenario. Matthias explained how a company approached him and the company he was working for as they were interested in the research he was doing in static analysis. They were interested in developing an application for static security testing and after almost half a year of negotiation the project started its development. There were two product owners (POs), one based in the same location as the client working within that organization and a proxy PO based with the same company Matthias worked in. They agreed on two-week sprints and at each review the team would show demos of the completed stories for each sprint.

Matthias' team started doing some internal training on Scrum and held a Lego-workshop similar to the one that was offered at the start of this course. They also did what Matthias called research spikes, during which each team member researched which tools and frameworks to use in order to develop the product. Even though this is less focused on customer value, Matthias thinks it contributed to evening out the workload and that it made everything less chaotic.

Matthias also presented the team members who worked on this project and described them as a "high potential team" given the qualifications of the people working on this project. However, even with the vast expertise among these members the development of the agreed upon product posed a real challenge as something similar had never been done before. Matthis then explained how the team quickly reached a working minimum viable product after two months of development, but quickly realized that it would not be viable to provide support for it. Several rewrites of the product and optimizations were carried out during the timeframe for the development, some of which included a 100x performance boost. Matthias elaborated on how these rewrites brought considerable customer value in the end, in spite of them implying starting the development of the product over again.

Moving forward, the general feeling within the team is that there were several differences between our experience in working with Scrum and what Matthias depicted. The consensus is that it would be best to identify these differences and select a subset of them to be applied in future projects to explore their benefits.

Some of these differences stood out more than others. For instance, the research spikes Matthias spoke about was an aspect that was never taken into consideration during the sprints our team had. However, it can easily be seen how this practice could contribute considerably to the development process and is something several members within the team have agreed to explore in future projects. Another difference that stood out was the double PO setup Matthias and his team worked with, compared to the simulated PO role our team had. In future projects it will be interesting to see how a non-simulated PO role works out, and having a proxy product owner working closely with the development team is something everyone agreed would be interesting to explore especially in cases where the work is done remotely and direct access to the real product owner is limited.