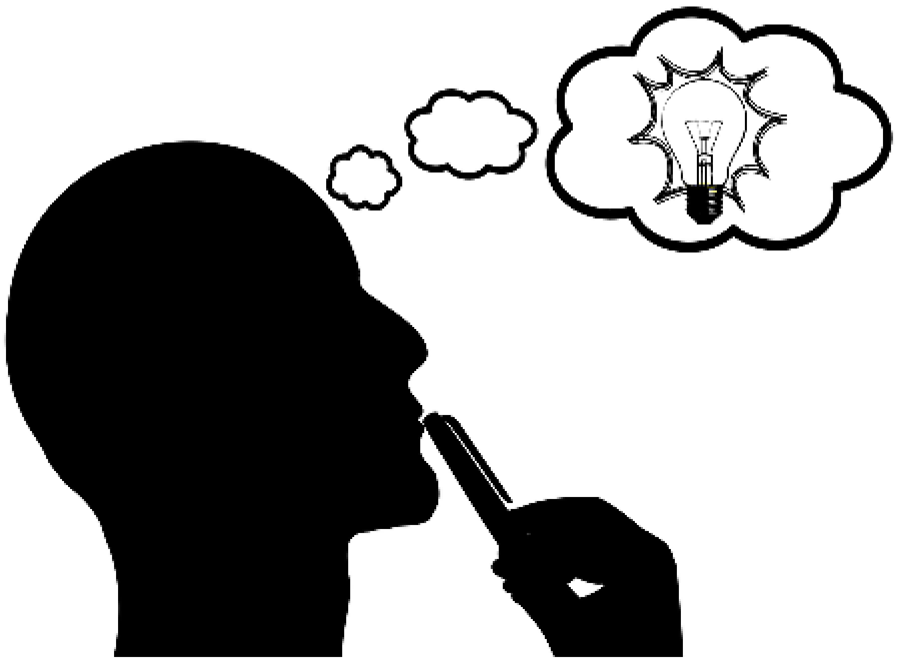
Reflection Report

Victor Dahlberg, Philip Laine, William Björklund,
Jimmie Berger, Fred Hedenberg, Edvin Tobiasson

October 2016



Contents

1 Introduction

We had been given the assignment to create an app for Android phones to motivate newly arrived asylum seekers to learn Swedish. We concluded early on that the best software for that would be some sort of game which, besides just assisting in learning the language, could be considered fun and entertaining. This idea was soon expanded on to instead become more of a platform for several games, including multiplayer, having a scoreboard, achievements and the likes to further motivate someone to study via the competitive nature of competing with your friends and reaching the highest score.
The development was to be done using the Scrum framework, which was well suited for the project due to the short time given from start to finish and to quickly iterate from an idea to an MVP and finally an end product. Even more so since we had an design team to brainstorm layout and ideas with, to which our goal became much easier to grasp if we had an MVP early on to present both to that team and other students to get vital input on how the app would be used and what improvements that could be made.

2 Application of scrum

The scrum method was utilized throughout the whole of the project, although to different degrees of correctness. Generally it was initially used as a shallow concept because of the requirement of using it, and later developed into something more purposeful. Certain aspects of working with scrum were never properly explored or taken into use by the group, but upon considering the issues we faced and reflected on them they've come across as tools that would in fact have been very useful during the project.

Roles, team work and social contract

The group didn't make use of specific roles throughout the project as such. No scrum leader, secretary or likewise was appointed. This never became an issue, but it's fair to say that the meetings could probably have been handled in a shorter amount of time if someone specifically took it upon themselves to direct the discussions towards the right areas. Considering how time became a commodity in short supply towards the end of the project it would definitely have been worthwhile to appoint a leader for the scrum sessions to more effectively walk the group through the different portions of the meetings.

In regards to the application itself the group members naturally became unofficial leaders in the sections they had spent the most time on. This was never explicitly discussed, but since the person in question knew the most about how the code functioned it became natural to discuss and handle changes to that section through them.

There was a good mood in the group throughout the whole of the project, and the struggles that presented themselves never seemed to create any friction between the members. The team work was lacking at times however, which is discussed more in depth in section 3, *Reflection on the sprint retrospectives* under *Daily scrum*, since that was supposed to be the main medium through which we structured our cooperation.

The social contract for the group was short and simple. It worked out well aside from the stipulation on always posting updates on how development was going, but this is discussed in detail under the above mentioned section *Daily scrum.* What proved the most helpful regarding the social contract was the commitment to meet in person every week. This was essentially a necessity since there was a project requirements to hold scrum meetings, but even if this was not so it would have been a good practise.

Used practices

The project was mainly carried out in a distributed fashion, that is to say group members working individually as they pleased on selected tasks. This was a very good thing in the sense that it allowed all members to contribute whenever they were able to, which is needed considering their different schedules. However it also proved weak in other respects. The main issue was in regards to communication and task distribution, although the later can be considered a result of the first. There were many times throughout the project when there was a need for continuous back and forth communication in regards to coupled pieces of functionality under development. A good example of this issue is in regards to the utilisation of the database. The different games were supposed to fetch different manners of data from the database to be used during a session, however there was a continual problem in making this work correctly. Both parties were certain they were doing things correctly, and only after explaining the motions of the respective implementations did it became apparent where the database communication was failing. Towards the end of the project this issue of communication became increasingly prevalent, requiring a proper solution.

Stand-up meetings started to be utilized. The group would meet and settle down for more or less lengthy sessions of coding and designing in the same space, allowing for quick communication and ease of explaining details. This was done with the whole group two times, and a few more times with fewer members present. It proved highly effective to implement pieces of functionality that were coupled in different ways, as well as for discussing and coming to agreement on different design choices. Another good aspect of doing this was that those present gained a better understanding of certain sections of the project in which they'd had little involvement, as well as generally making everyone better updated on progress. This practise would have saved a lot of time and effort if it had been considered from the beginning, and if the project was to continue on it would have been used continually to clear troublesome portions of development.

Time distribution (person / role / tasks etc.)

The time put into the project was nominally equal for all group members. Since there was no particular role assignments within the group there was no special allocation of time in regards to that. Time spent was therefore more or less equalised for all members. Time spent on lectures is skewed towards certain group members, which will have to represented as an element in the individual scoring the group members give to those in the group.
The time spent amounts to:

- Working on tasks
  The group went through two 5-day sprints and two 2-day sprints. This amounts to a nominal time of 96*2+38*2=268 hours.

- Participating in lectures
  This wasn't kept track of perfectly, but assume two group members participating each lecture. This amounts to 9*2 sessions of 1 hour and 45 minutes. A total of 31 hours 30 minutes.

- Design meetings
  The group had two meetings with their designated designer, for two hours each. Total time becomes 4 hours.

- Other
  There were five large course elements which all group members participated in: *Lego exercise, project introduction etc.* One or two of these may have ran an hour short, but a nominal estimate is: 4*4+3(lego exercise)=19 hours.

Total: 322 hours and 30 minutes.

Effort and velocity and task breakdown

At the start of the project there was no attempt to apply the concepts of effort and velocity. A task was thought out and loosely assumed to be of a reasonable workload for the time frame at hand. These tasks were also quite large and horizontal in scope which led to different issues, more on that matter can be read in section 3, *Reflection on the sprint retrospectives.* In the review of the first week it was noted how most of the tasks had been improperly assessed in regards to the time needed to finalize them. At this point the group started to delve into actually utilising the concepts of effort and velocity from the seminars.

Velocity

In the seminars a four hour workday had been proposed and the group's velocity was calculated based on this. A typical sprint would span from Friday to Friday, making it five days in total. With a group of six this would equate to a velocity of 120 hours. There was no assumption however that there would be any unexpected loss in productivity, be it from other courses demanding focus, sickness or other reasons. A constant of 0.8 was decided on to counteract this and make for a more realistic velocity for the sprint. A typical sprint would therefore equate to *5*6*4*0.8=96 hours.*

During the second week it was decided to hold a short sprint consisting only of Monday and Tuesday, leading to a velocity of 38 hours. This was done due to an upcoming meeting with the design team and a wish to bring a further develop application to allow for more helpful feedback. However in the sprint reflection of this second week it was noted that such a short sprint was harder to successfully accomplish. This should not be the case, but was asumably due to less than optimal working practises as well as the presence of other courses. There was a tendency to put off a day's work and then putting in that effort at a later day. Sometimes this could be because of laboratory work or other taxing portions of other courses and other times because of less prudent reasons. Naturally this was not feasible in the same manner with a sprint lasting only two days. It was decided that such short sprints would be avoided in the future and all sprints after this were five days in length. In retrospect this seems correct due to the nature of the group members studies demanding different prioritization at times. With this said there was one more instance of an essentially two day sprint, which was in the last week prior to the final presentation. This can be considered sensible however to try and assure quality.

Effort

The effort of the tasks themselves were approximated by the group. This had been suggested in the seminars to be done by each group member writing down their suggestion, and from there find a sweet spot. The group did this less formally. A suggestion in regards to the tasks effort was raised, typically by the group member who came up with the task. Others would then agree or disagree and a decision would be reached. This process was never discussed further during the project. In hindsight however, the suggested method of using cards to write down suggestions would probably have been preferable. There would have been less of an opportunity to simply accept someone else's suggestion rather than truly considering it. By having more opinions one can assume that a better estimate of effort could have been reached. Were the project to have continued further from this point this practise would have been implemented.

Task breakdown

Due to our tasks being user stories, the way they were broken down and worked with are covered in section 3, *Reflection on the sprint retrospectives.*

3 Reflection on the sprint retrospectives

At the sprint retrospective meetings, many different aspects of the process were discussed. Two of the most troublesome areas throughout the project were *user stories* and *daily scrum.* Because of this, they will be covered separately.

Another area covered during sprint retrospectives was communication. In a high paced project such as this one, communication cannot be stressed enough. The group usually drifted apart, sometimes for a few days, in terms of communication. Sprint meetings helped getting us back on the same track in terms of where the project was going, but it was not enough by far. This issue is highly correlated with *Daily scrum* and actions taken to address it are discussed there.

Some of the deadlines, such as demoing and final presentation, were not aligning with the Friday-Friday sprint schedule. To finish user stories before the demo, the sprint was cut short by two days (together with a corresponding velocity). It was later concluded that it was not a good practice. There was not enough breathing room to do other things, such as focusing on something else for a day and do eight hours of coding the next day. For an eight week project, a seven day sprint is probably a good practice. Especially if the group is not physically located together while doing programming, since this might lead to the group drifting apart in terms of communication. With a shorter sprint, this is naturally addressed.

User stories

The group's initial thoughts regarding how to structure and define the different tasks of the project evolved throughout the weeks. Initially the terminology of a user story was only used as a shallow concept. The user stories we drafted were horizontal slices of functionality. They were essentially justified as user stories by the use of formalised language, written as a request by a user but with poor thought regarding the content. This is a loose manner of working on a project that has been sufficient through many other student projects the group members have participated in. As an example, one of the user stories from the first week was: *"As a user I'd like to be able to play with my friends."* This essentially amounted to implementing multiplayer functionality, which is both massive in scope as well as a highly horizontal task. With the requirement of continually providing value which needed to be fulfilled throughout this course however it proved inadequate.

The issue was that tasks became too large and they encompassed too many different elements. Being too large meant that it was difficult to judge whether it was really possible to accomplish in a given amount of time. Even if a task seemed manageable there could be unknown pitfalls that would increase the actual effort needed to finish it. Firstly, if this happened, no value could be presented at all. Secondly, the fact that one single user story encompassed so many different manner of tasks made it difficult to work as a group. It seemed natural to work on different parts of the task, but being encompassed in one user story made it hard to draw a line in regards to what each individual should work on.

After the first week some of these issues came to light in the retrospective and an attempt was made to better apply the carpaccio method from the seminars to our user stories. Limiting the scope of the tasks was a struggle, as was estimating the effort associated with them, but the process was iterated upon through the different sprints and improvements were made.

However in many cases it still felt too limiting to implement such small user stories. Therefore a new design in which the user story itself was to a differing degree horizontal in scope was thought of. This story would be made up of multiple sub-tasks which would be vertical slices in regards to their contents. Essentially multiple pieces of functionality which belonged together were gathered and placed under a horizontal user story with individual effort values. At this later time it might seem obvious, but this way of working was prone to many of our original issues. It was not a complete relapse in methodology since the creation of more defined sub-tasks lead to a greater effort in considering the effort and verticality of the tasks. The user stories created in this manner did not necessarily suffer from being hard to achieve in the allotted time, but they still proved hard to cooperate on. Since these user stories were typically the main creation of one group member, the sub-tasks typically connected in some loose manner and followed a certain flow. There was a certain feeling that you might be encroaching on what someone else was doing. This made it awkward for others to work on different sub-tasks in the story. An attempt at solving this was made after consideration at the third sprint retrospective. The group gathered in the same workplace despite working on different tasks. This allowed for quick communication which made it much easier to realise what was appropriate to work on. Although creating an environment such as this proved effective, it essentially served as a band-aid. It allowed much less flexibility in regards to when and how group members worked on the project.

With the different attempts at handling user stories and the experience earned from it, we feel that there is definitely a practical value to doing so in a well considered manner. We did handle certain tasks as small, solitary user stories with vertical content, and these ones never turned out poorly. We feel that we should have continued to try and break up all our tasks to align with creating user stories like this. The creation of user stories with sub-tasks which at the time felt like a clever compromise proved to be half a step backwards. The experiences from this project were enlightening and showed the value of the advocated carpaccio method, something which should prove helpful in the future.

Daily Scrum

At the very beginning of the project, daily scrum did not come very naturally. Since all group members had different schedules and some lived far away from campus, it would be ineffective to have a place and time to be everyday, due to a it being a short ten minute meeting. The initial daily scrum system was that everyone had to send a message in the project chat (a Facebook messenger group) about the days work. Unfortunately, these daily scrum messages quickly disappeared in long discussions about other things. This lead to group members not having any idea of what the others were doing. As a compromise, another group chat was created, specifically dedicated to daily scrum. Moreover, during Monday to Friday, everyone was to write what they had done and what they would be working on later that day. What time during the day did not really matter. Organizing daily scrum like this resulted in a lot of flexibility.

This way of doing daily scrum was practised throughout the project. In the middle of the project, the system worked well. Everyone participated and wrote their part in the daily scrum. On the other hand, during both the beginning and towards the end of the project, it did not work well at all. During two sprint retrospectives, this issue was addressed and actions were taken. During the last sprint retrospective, the group realised this system would not work out in the long run. A few different ideas on what actions to take were discussed:

- Have a fixed place and time. Either a meeting at campus or using group video calls
- Fewer daily scrums, for instance three times a week
- Have it embodied in "drop in"-days, where everyone can drop in whenever they want during some set hours

There was a paradox in that when the group sat down in the same room to code for a few hours, we *had the chance to have a real face to face daily scrum meeting*, but since everyone heard what everyone else was discussing, *there was no need for one.* You naturally got to hear the things everyone else was working on anyway. This also serves as a good example of how impactful coding in a group is, especially regarding communication and vision.

4 Documentation of sprint retrospectives

The sprint retrospectives were during the process of the project referred to as "sprint reflection". All four sprint retrospectives can be found in the root folder of the Github master branch. Link: https://github.com/hardvaluta/android/tree/master

5 Reflection on the sprint reviews

Sprint reviews were performed simultaneously with the sprint retrospectives throughout the project. However it was touched upon very lightly and was not given sufficient consideration, which made the quality of the application suffer at certain points.

Through all the sprints save the first one, the current state of the application was considered good and that the assigned tasks had been completed. This was accepted despite not fully considering if it was the case. The basic, main functionality was typically there, but other important aspects could be missing without this being noted. An example of this is how the first game 'Finn rätt ord' was presented for feedback on a design meeting with a bug present which caused crashing. The bug was easily triggered through play, which means that it really should have been caught.

A way to remedy situations like this would have been for the group to properly define what it means for a task to be 'finished'. There should have been standardised steps to take in regards to whether a task was complete. It did not need to be complicated, and an example checklist could be:

- 1: Is the stated functionality present?
- 2: Is it visualised in a decent way? (Changes can be made later, but it should still be acceptable and made with thought)
- 3: Any obvious bugs noticed through normal usage?
- 4: Try to check for bugs with typical 'strange' usage. For instance spamming a button, pausing and resuming the application and more.

The fourth point could naturally be broken up and defined better into certain steps, but a list to follow such as this would have served to improve the state of the application after each sprint and therefore ascertain the value that was delivered. Had the project continued this list would have been taken as a starting point that could later be iterated on.

The scope of the project is also something that should have been better considered. During all sprint meetings the group discussed which steps that needed to be taken to be able to deliver our finished application as we had envisioned it. There was never any thought raised in regards to whether we would be able to finish this on time. It was simply assumed that it would be completed. Ultimately this was not the case, and the feature of visualising a player's score was strongly underdeveloped.

What should have been considered, and acted upon, are two different things. Firstly is that features considered important should have been prioritised as such, making certain that they would be properly implemented in the finished product. Secondly the features that were not properly completed on time should have been cut from the application rather than be presented unfinished. Had this been done the final product would have been markedly improved.

6 Best practices for using new tools and technologies

Using Android offered some challenges as some members in the group had no prior experience with it. Time needed to be spent understanding the inner workings and style guides of android as the project relied heavily on the app. Both Android and iOS have their own ways of accomplishing the same goal, which meant that something done on one platform was not applicable on the other.

This caused some issues as there was no shared design language between each group member. As everyone had a different background and experience with the OS. This meant that some extra time had to be spent on redoing the design to fit the the android stylistic language. The challenge was to have a design that would work with our application while at the same time keeping it intuitive for long time android users.

7 Reflection on the relationship between prototype, process and stakeholder value

During this project the team obviously spent the most time on the prototype and the process correctness suffered from it. When the pressure was on and the prototype was prioritized it is ironic in retrospective that the product would probably end up better if more time was allocated to the process. The prototype could not exist without the process because without process there is no vision or purpose. Whatever prototype you end up with its just meaningless code if it does not serve a purpose. Purpose is provided by the stakeholders/product owner and is the basis for a good vision to base the process on. The stakeholder value of the product was continuously investigated during the project, mostly through the eyes of an hypothetical intended user (a person with minimal knowledge of the Swedish language) but also with help of the interaction designer. The stakeholder value was defined and prevalent all through the project overall but could have been more clear. The stakeholder value should be clear in every aspect of the prototype where it is possible, but this was not always the case. All things should have been designed with a purpose and not have a purpose added afterwards. This rarely happened but sometimes code was written or designs made more for the sake of getting things done instead of being more deliberate. Although this could be viewed as bad practise it helped creating ideas and features not previously thought out. A project must be viewed as a living entity and things can change during its lifespan. New forms av stakeholder values can arise, maybe from opportunities created by the prototype. The team learned during this project that it is important with a vision but what you set out to make and what you end up making will not necessarily be the same, and this is not a bad thing but should be viewed as a part of a healthy process. Just as product owners might change their mind and that can lead to changes in the prototype and process, developers can present things through the prototype that can have a stakeholder value even although its not initially specified.

8 Relation of your process to literature and guest lectures

Throughout the course, on average there would be two group members attending the lectures. The ones attending would then sum up what the important lessons learned and share it with the others. Either by explicitly telling everyone about it or by simply taking on tasks where the lesson learned can be of use. In other words, what was taught at guest lectures (as well as lectures by Håkan or Jan-Philipp) was continuously implemented and used in the process.

Before the first sprint started, all group member were to learn some (or refresh) android in general, android studio, Github and scrum/agile. No actual curriculum was established. Instead, everyone was to find their own ways of learning. There are a number of good and bad aspects to this practice. By letting everyone find their own sources of knowledge, a diverse knowledge base is created. Several different ideas on what a certain word means can arise. For instance, what is Scrum and agile? How is it deployed? Different ideas can be good, but they have to be discussed in a group. Otherwise the group might end up working in a counter productive manner, since there is no real consensus on what something means or what is to be done. Therefore, if a group is not communicating their idea on what something means, it might be a bad practice to not have a set curriculum. On the other hand, if the ideas are in fact is discussed in a group, a diversity of opinions can be a contributing factor to a successful project. During one of the sprint retrospectives, it was concluded that scrum terminology and practice was not too well known within the group. Therefore everyone was to read about it over the weekend and afterwards discuss scrum in general and what integral parts there were. This concept about diverse ideas and the need to discuss them also applies well to the product vision. To create a common purpose and align the group with each other regarding what the product vision means is essential[1]. This is something that needs to be discussed thoroughly within the group. Without a common idea about what the product vision means, especially with such a sparse communication as in this project, it would be very hard to deliver a product everyone feels good about.

Throughout the project, the product vision stayed the same. What changed was the different features and parts that would add up to a product satisfying the the product vision. An important lesson learned was about ability to adjust and change. "*Since we don't know exactly what we want we strive to do as little as possible, but do it in a way so that we can get customer feedback on it, even if we have to do some rework as we learn more. This pays back in the end*"[2]. This quote sums up how the group used an agile process to develop a product. With a product vision instead of product specifications, the group was working towards a not so well defined goal. Sprint after sprint, the goal gradually became more clear.

9 Evaluation of D1A and D2

During the course, assignments were presented to the team with different purposes. The first one emphasized on learning scrum and getting some good practises started. The second one was a half-time review where the team was to evaluate the progress, practises and overall work that had been put in.

D1A

There where mixed opinions in the group regarding the scrum exercise. The team learned from it and still feels it was a good way to kickstart the application of scrum in the project. Unfortunately the group concluded that the effectiveness of the exercise was somewhat undermined by its own design. It seemed like there was a point to be made using shock value and limited information about the assignment, this was confusing and helped drive the impact of the realization home. But afterwards the opinion was that the scenario was a bit unrealistic and that weakened the assignment. In retrospect one can say that there absolutely was lessons learned from the assignment. But as is evident by the groups somewhat poor scrum practices obviously a greater effort to incorporate scrum should have been made. That stated, the team still believe that the application of scrum would have been worse without this exercise and even although the team members appreciated the task differently, all learned from it.

Looking back at the written part of the assignment where three insights was jotted down they all seem obvious now which is a good thing. The simple realization that it is very easy to have several people or groups to work on the same task separately without communicating and how badly that affects the producing power in a project is huge. Just to be aware that there is a stakeholder/product owner/intended user affects the work-flow immensely. The assignment taught the group how to come together as a team and get into the mindset to develop for someone else. The mindset that there is an intended user is very helpful for developing. More feedback could have been taken in from actual humans and it is unclear how of it would affect the design.. It was regrettable that the visit to an refugee center that the interaction design students had planned did not happen. Since this did not happen an effort to orchestrate something similar should have been made by the team. This is something that could have been done better and would have been given a second opportunity. From this it is learnt that if an external source fails to deliver one should try to get it done by other means if there is a way.

Something that the group did stay true to was the decision during this assignment to use modular programming with the primary purpose of having an early MVP. This was accomplished and provided a good platform for continued development. Having an early MVP also made it easier for the interaction designer to provide feedback and suggestions for the design and the same goes for the team itself. The early prototype also led the group to be able to have thoughts on the products design early on. The choice to program modular from the get-go also made it easy for the group to work on different parts of the application simultaneously.

In closing it is obvious that the assignment was effective and helped the team to start the scrum process. The problem that rose in relation to scrum had more to do with the teams implementation of it and a general lack of discipline in this particular area.

D2

The main subject of the half-time review was the poor scrum practices the team had shown. Applying scrum and being vigilant about defining effort and velocity got pushed aside by the efforts made to have an early MVP. The problem with scrum was based in poor routines and lack of discipline, the remedies where simple in theory and worked out well initially. The problem with effort and velocity stemmed partially from it being hard to appreciate such concepts early in the project. Without a lot of prior experience the team found it difficult to put effort on tasks and harmonize it with the velocity. But the bulk of the problem is simply that a correct process was not prioritized compared to getting a working prototype. This was a bad idea coming from a lack of experience. It was deemed more important to get code in the repository than having a clear vision for the product and using a correct, well thought out process. This was a clear lesson learned in itself since problems rose from this lack of planning and structure that could have been avoided. After the demonstration a lot of the teams process flaws where exposed. Looking at the half-time review one can see that the goals where maybe met each week more or less but the way of getting there was not very clear. Even though the MVP was there the work of estimating effort and velocity had not improved much and obviously the group had problems grasping the value of the process at this stage.

A lesson taught is a lesson learned. With the product not fully finished and some features lacking at the moment of demonstration, it is a clear example on how one should always strive to implement a well structured and correct process.

On the upside a working prototype existed and allowed for constructive criticism now that the interaction designer had been integrated to the project. This maybe should have been done earlier, having the wire-framing concept introduced earlier could lead to a good design being implemented earlier. Discussing and talking through the flow of the application would have been very useful even with next to no code written. The workshop solidified many of the existing ideas, changed some and eradicated a few. Getting a grasp on the design was immensely helpful to creating smarter tasks in the backlog which was sorely needed. Having smaller slices (carpaccio) would make it easier to prioritize tasks and track progress. For future projects it would be wise to cement a vision and discuss design early to make dividing tasks and checking them off a lot easier.

The concept of scalability was still prevalent within the application and made some the work much easier despite the lack of intelligently created tasks and proper priorities. Overall the work had proceeded at an acceptable rate. Solutions for where the team was lacking was proposed and the half-time evaluation was concluded. Since this was practically the first time working with a backlog a lot was learned simply by trial and error. Understanding the value of having a backlog and having a somewhat successful implementation of it will greatly help the next time one would work in this manner. Experience providing a clarity of vision and an understanding of the scope.

To sit down at a checkpoint and actually reflect on what has been done, what there still is left to do and what could have been executed differently is a good way to evaluate progress. Being able to acquire an overview of the project and zoom out helps clarify what to prioritize. Regrettably, even though the problems where pinpointed during the half-time evaluation and initially remedied, the teams scrum practises deteriorated during the last part of the project. This was caused by the stress of getting the product ready for demonstration. Upholding good practises while under pressure is a mark of experience and from completing this course one could assume that similar situations will be handled better in the future due to gaining that experience.

10 Summary

The two goals of utilising the scrum method as well as presenting a finalised product both fell a little short. What is interesting to consider is how the two are closely coupled. Throughout the different sections of the report it has been repeatedly noted how the issues in regards to the product's state often could have been avoided with a better usage of the scrum method. Doing so could have ascertained better quality and better usage of time that could in turn have been spent fully finishing more features. A related point is that if more effort had been put into the scrum process and truly looking ahead at the time that was left to us, then important decisions of cutting content to create a better product could have been made. Agile processes in general can cut down on amount of early decisions needed. In the world of software, teams need to be quick on their feet and adaptable to the everchanging conditions of the project.

The group is in agreement that the project has been very enlightening to work on in many respects. Perhaps most notably it has been eyeopening to consider what went awry in regards to both process and product, and noting how there is a connection. Many aspects of the scrum process can seem unnecessary or like wasted time. But by working with it and reflecting after the fact, the underlying reasons for many of the steps outlined for scrum have become clear. Likewise it is now much more understandable why many companies have incorporated this method into their work flow. By utilising scrum, companies are able to reliably deliver value to their product owners continuously and receive feedback on it.

11 References

[**1**] Roman Pichler, ”The Product Vision”.
”https://www.scrumalliance.org/community/articles/2009/january/the-product-vision”.

[**2**] Mario Carlsson,
Volvo Cars. Guest lecture, ”From Waterfall To Agile”.