DAT255

Software Engineering Project

Reflection report (Group Stierna)

Participants:

Debesay, Petros (petrosd)

Holm, Jesper (holmje)

Lahti, Gustav, (lahtig)

Linderholm Wamby, Magnus (wamby)

Logren, Emil (loemil)

Ngo, Tuyen (tuyen)

Palmberg, Paulina (paupal)

Runvik, Arvid (rarvid)

Ruud, Lucas (lucasr)

Thell, Hannes (hannest)

Abstract

The report describes the different parts of Scrum and how the group implemented these in our project. There were a lot of setbacks and difficulties throughout the project. These have been described and evaluated with the intention of finding out what could have increased our efficiency and led to a better end result.

Contents

1 Introduction

This report will detail and discuss the work methods and applied practices of the team Group Stierna in the MOPED project. There will also be a discussion of what could have been done better, and what steps could be taken to implement better work methods for the team in similar projects.

1.2 Background

In the field of software engineering, agile work methods are often implemented as a way to handle complexity and to increase the overall quality of products. Good knowledge in the theory behind these methods, as well as practical experience in working with them, are both important parts in preparing students for working within software engineering.

1.3 Purpose

The purpose of this report is mainly to reflect on the work methods that the team implemented in the MOPED project, in relation to what could have been done in a better way. There will also be a discussion regarding steps that could be taken to improve the workflow in a software project. The purpose of this report is not to detail and discuss the technological aspects of the MOPED project.

1.4 Problem

The problems that the team interpreted the MOPED project to handle are as follows:

- How to plan the workflow of a software project
- How to implement said plan to produce stakeholder value and a prototype
- How to implement Scrum and how to adapt Scrum according to the needs of the team
- How to reflect on a chosen work method and how to adapt said work method according to the present circumstances (such as technological obstacles, new tools, inefficient workflows etc.)
- How to define the relation of a process, a prototype, and stakeholder value in a software project

1.5 Obstacles and Limitations

The most prevalent obstacle for the team was the limited practical experience the team members had with Scrum and other agile frameworks. Previous experiences with software projects and similar types of work had focused much more on the technological aspects of the assignment. Therefore, sufficient skills in project planning and agile work methods were lacking

at the start of and throughout the project, which became clear in the inefficient workflow and lack of organization of the team.

The team also lacked experience with hardware. This drastically increased the time required to solve various hardware-related issues.

2 Chosen Methods and Reflections of Methods

This section will provide a description and reflections of the work methods and applied practices that the team implemented in the MOPED project. The foundation of the team's work method was Scrum, which was adapted according to the team's prior knowledge, experience, and perceived needs.

2.1 Social Contract

The team had a written social contract that was agreed upon before the start of sprint one. This included promises regarding implementing daily Scrum, active participation from all team members, transparency from all team members, and agreement on a general workflow.

The social contract was however mostly disregarded and not actively applied after its creation.

A possible explanation to this could be that the team did not feel the terms were realistically achievable. For example, the contract stated that "The group members shall be on time for all meetings set by the group", but after a few tries the team realized this did not reflect reality. The team members solved it by informing the rest of the group when they would arrive and the reason to why they were late. The solution worked, but it should have been written down in the social contract to better the terms, and make them more achievable.

2.2 Roles

The only formal roles of the team were Scrum master and developer. The Scrum master only did work related to the role part time, and worked mostly as a developer. The developers participated in all types of work related to the implementation of the product, as well as adjacent work, such as sprint planning, reflections, and documentation.

The role of Scrum master was not implemented well enough to keep the team working according to Scrum all the time. Since the Scrum master mostly worked as a developer, there was no one in charge to make sure that Scrum was followed.

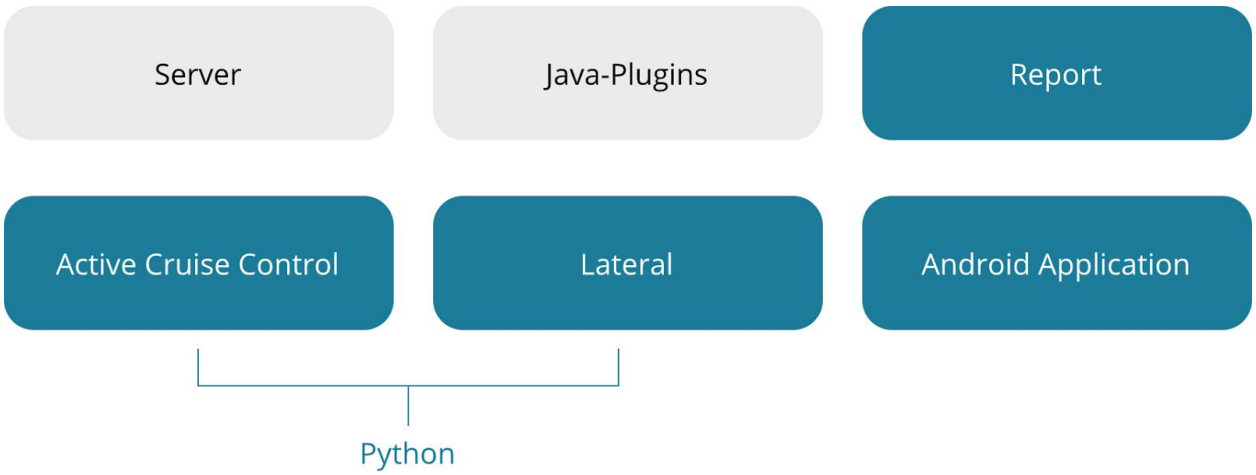
It is important to point out that the team members lacked previous knowledge about Scrum. Not having a fully dedicated Scrum master could very well be the cause of the inhibition of functional Scrum. Due to the inexperience, it was needed to have someone telling the team what and how to do in order to get most of the benefits of Scrum.

Having someone assigned the project manager role could have also helped the workflow and project structure. It would have been a good addition to a Scrum master, whose sole responsibility would have been to manage Scrum.

2.3 Teamwork

There were no set rules on how the teamwork should be, but active participation from every team member was agreed upon in the social contract.

The group split into separate, autonomous sub-teams when the problem areas allowed for it, each handling its own problem area. Overall, coordination of the team did work well but with varying success especially towards the end.



Different problem areas. *The grey ones are the ones that have died out as the scope changed.*

Ideally, the team should be divided into smaller teams on a task-by-task basis. This increases parallelization which in turn increases the efficiency. These smaller teams should also communicate with each other to make sure that everything is going well, and if needed, borrow a team member or two to make sure the sprint goals are achieved.

The teamwork could have been improved by setting rules and standards for how the team should be divided into subteams, how these groups should work, and how these groups can change over time. To make sure these rules were enforced, one person, e.g. the project manager, should have been responsible for dividing the subteams and changing them if needed, though this role could potentially be phased out as the team became more experienced.

2.4 Used Practices

The team strived to use many practices, but since this is the first time for many team members to ever use them, they were not properly utilised. As described below, the team did not take full advantage of the used practices. This could be explained by various reasons, and they will be elaborated further in each subchapter.

To begin with; the team is incredibly inexperienced in comparison to the guest lectures that were given by the companies IGDB, Zenuity, and Volvo Cars. IGDB shared that they had to learn the hard way to get to their best practices by trial and error. Volvo Cars share a similar picture. During their lecture, they showed their best practices – what works for them. This is something that has required much time (years) and effort to get to. In comparison, our team has had about 5 sprints to figure out what practices worked for us, with few reference points.

However, it is fruitful to now analyse what went well, what did not, and how to improve such practices so that the team can benefit from them. It is a positive spiral, once you see and experience the benefits of the practice, you will use it more and refine it even more to yield greater benefits.

2.4.1 Scrum Board

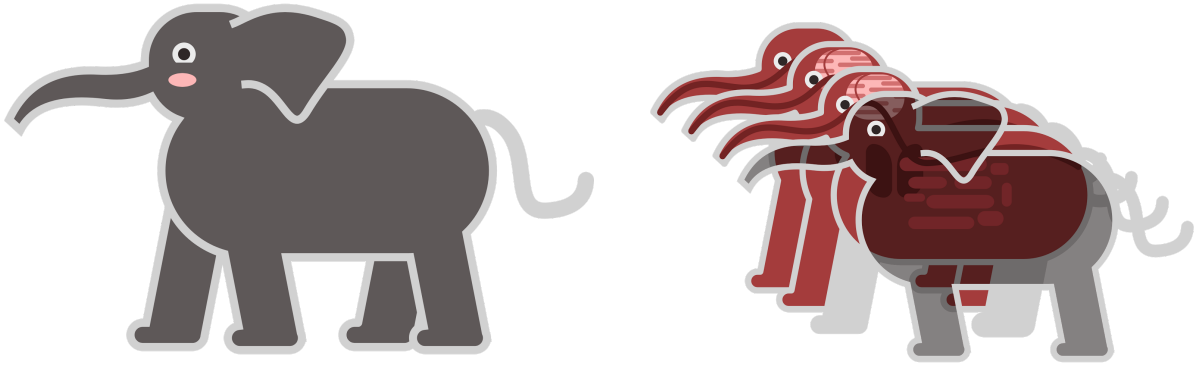
The software Trello was used to keep both the product backlog and the sprint backlog on a single board. The product backlog consisted of three lists: high level user stories, broken down user stories, and tasks. The sprint backlog only consisted of tasks, with lists showing which tasks were in what stage of implementation (to do, in progress, testing, done).

Product backlog. *The cards in Product Backlog (Tasks) were moved, depending on their status, to similar lists with the headings To Do, In Progress, Testing, and Done during the sprints.*

This method of dividing the work effort was a direct result of the elephant carpaccio exercise, which showcased the large improvements that can be done to a traditional distribution of effort. The prime principle is to design specific problem areas from the view of a customer in a vertical manner so that the customer can see the fruits of the labour immediately. Following that logic, several high level task were reshaped to be more “vertical”, so that they include all levels of development. For example, developing a complementing phone application changed as following:

“developing network communication” and “developing user interface”	→	“develop connection process” and “develop steering process”

The positive aspects of vertical slicing was especially apparent in some parts of the project, as it was a confined development feature which could incrementally be built and added to. However, the majority of the project required more investigation and research than development, which is naturally harder to develop and showcase to stakeholders.



Elephant carpaccio. *Slicing the poor elephant vertically shows the product owner that they will eventually get their elephant.*

The main usage of the Scrum board was in planning sprints and prioritizing backlogs. The board served as a visual tool for what the sprint would consist of and how much work would be done on which problem area. The latter was achieved by using tags, a feature which lets the team associate an item with a label, e.g. a problem area, such as “lateral control”.

During the sprints, the Scrum board was not used regularly, and was not a part of the team’s daily workflow. Therefore, tasks were often not assigned to individual team members, nor

moved to the appropriate list when the status of the tasks changed. Because the Scrum board was not used regularly, the benefits from it became fewer and smaller, which further decreased the usage of it. When the low usage of the Scrum board was noted by the team, efforts were made to increase activity, and the perceived benefits of the Scrum board increased.

The general consensus regarding this practice is that the team needed to be more active with it. Instead of mostly disregarding this tool, the team should have assigned the tasks to different team members. As a result of this, efficiency would increase noticeably on its own. The team should also have added or removed tasks, depending on how the situation was during the sprint. Lastly, the group should more actively have moved the tasks in accordance with their statuses. None of this was being done on a regular basis. This could have been fixed by having a full time Scrum master as well as more concise tasks and specific tasks.

2.4.2 Pair Programming

Pair or group programming was used throughout the project in a few variations. In the beginning, it was done in bigger groups in order for everyone to get an understanding of the MOPED platform and because of a lack of problem areas. This was reasonable at first as the purpose was not only to output code, but also to get a better understanding and a common ground to stand on. It was quickly realized that smaller groups were necessary to achieve higher efficiency. As the team created more structured sub-groups during sprint 3, the programming was done in groups of max three. More specific tasks and better structure for sub-teams would have made it easier to structure pair programming and would have increased its efficiency.

2.4.3 Daily Scrum

Daily Scrum meetings were not held daily, nor with the participation of every team member for every meeting. When technological implementation was hindered, the team did not feel that the meetings added any value, as there was often nothing new to report. For most sprints, about three daily Scrum meetings were held, with later sprints averaging higher than earlier ones.

The aspiration of the team, as stated in the social contract, was to conduct daily Scrum meetings for which every team member was present in person. These meetings would consist of every team member sharing what they had done since the last meeting, what they planned to do until the next meeting, and if they had any issues or planned to do any work that might cause issues for other team members.

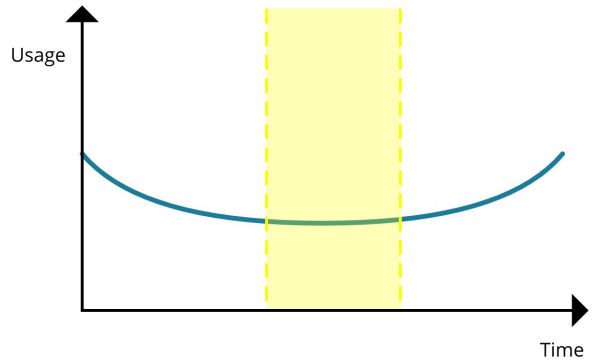
To conduct the daily Scrum, the team (and especially the Scrum master) should have decided on specific times and places to hold a short daily Scrum meeting preferably at least one day in advance.

2.4.4 KPIs

Another practice that was applied was keeping track of three KPIs, which were believed to indicate significant quality of different aspects of the project and work methods. These KPIs were:

- Estimation Accuracy
- Happiness
- Git activity

As a general trend, the measurement and usage of the KPIs followed an inverted bell curve, with less frequent usage towards the middle of the project. The exact values will be presented in the charts below with an accompanying discussion.



The KPI usage over time. *The usage dips in the middle of the project (see selection) as more focus is directed at the implementation of the project.*

2.4.4.1 Estimation Accuracy

Estimation accuracy was determined as a quotient of the total achieved effort of a sprint and the velocity of that sprint. This would show how close to the estimated velocity the team actually came, and would be used to better plan sprints in the future. The team used a burndown chart to keep track of this KPI. Initially, the team based the effort points with time on a 1:1 ratio, i.e. 1 hour of work is 1 effort point. However, it was not the best method as it was hard to accurately estimate the tasks based on time.

Since the estimation accuracy was dependent on the burndown chart, it was important to fill in the completed effort carefully every day of the sprint. Unfortunately, the team didn't do that well. Team members often had to remind each other to fill in the burndown chart, so it was usually filled in a retroactive manner. The chart that follows shows the team's accuracy.

Sprint	Accuracy (Total effort/Velocity)
1	0,64
2	0,79

3	0,99
4	0,45
5	0,82

The accuracy values did change from week to week. At first sight there is no logical pattern to identify. However, by reading each week's sprint summary it's easy to understand why the desired accuracy wasn't achieved.

During the first sprint, none of the team members had any experience of estimating effort needed to complete a task. In the second sprint the group had better knowledge of estimating the accuracy and therefore gave a better estimated value. In sprint three the team was divided into subgroups and an almost ideal accuracy was achieved.

In sprint four and five the accuracy did deviate from the previous ideal accuracy. This was a result of limited access to hardware. At that state, the majority of tasks which involved testing were ready to be tested, but could not.

2.4.4.2 Happiness

Happiness was measured with a weekly survey, conducted at the end of each sprint. The team members would answer the questions

- "How happy are you with our product?"
- "How happy are you with our team?"
- "How happy are you with our workflow?"
- "Do you feel like you have contributed enough?"
- "How stressed are you?"

on a scale of 1-5. An answer of 1 meant a bad score, and 5 was the ideal score. The total average, as well as the average for each individual question would be summarized each week, so that the team could spot negative trends in the different areas that the questions covered individually, as well as overall happiness.

This KPI was the easiest and most consistently kept track of. As shown below, the average was consistent throughout the project. However, due to the large amount of data points, significant differences were likely hidden by only looking at the average.

Sprint	**Total average (out of 5) 1 = Not Ideal, 5 = Ideal**
1	3,56

2	3,22
3	3,56
4	3,52
5	3,24

As can be seen, there is not much variance in the numbers, despite the change in the mood being very noticeable. This shows the limitation of the measurement of this KPI. There was a much higher variance for each question though, e.g. how the stress level changed from sprint to sprint. Therefore we can conclude that instead of comparing the averages of all questions, which tells us that nothing went wrong, we should instead look at mean or individual values and address those directly.

2.4.4.3 Git Activity

Git activity was measured by looking at a number of factors in the git-repository. The chosen factors were:

- Number of pull requests to master
- Number of merged pull requests to master
- Number of commits to master
- Quotient of merged pull requests and total number of pull requests to master
- The stability of master
- The difference in activity on master since the week before

The hope with this KPI was to show activity in matters related to version control, to indicate whether enough was being done on that area, which was deemed extra important considering the size of the team.

Measuring git activity did not require much time and effort as it was auto-generated. These numbers were rarely used during the sprints to re-evaluate our work.

Sprint	**1**	**2**	**3**	**4**	**5**
No. of pull requests to master	1	1	1	1	1
No. of merged pull requests to master	1	1	1	1	1
No. of commits to master	4	4	9	10	10
Master stability	N/A	N/A	159.2	159.2	159.2
Pull req. quotient	1	1	1	1	1

No. of Δ pull reqs to master	1	0	0	0	0
No. of Δ merged pull since last week	1	0	0	0	0
No. of Δ commits since last week	4	0	5	1	0

These values did not provide meaningful data, as, up to the last sprint, very few features were developed enough to be added to the master branch. Most of the time multiple unconnected tasks were on the same branch, which meant that a branch could not be merged with master when a specific task was finished.

Each task, group of tasks, or user story should have its own branch so that when it is finished, it could be merged with the master branch or a de facto master branch for a specific group of tasks or user stories.

To accomplish this, a well defined and proper git-strategy should have been formed. To make sure that strategy was realised, the team would have needed a team member that enforces this strategy and teaches the rest of the team to follow it when needed.

2.4.5 Effort, Velocity, and Task Breakdown

One of the greater problems during the project was the size of the tasks. This made it difficult to terminate all tasks in the set sprint. The tasks were rarely moved to the correct lists (with the different states e.g. In Progress), and many times they were not finished before the sprint ended. This is solely because the tasks were too big. An example of this is the task “Integrate ACC with lateral control so that a reasonable speed is kept when preceding vehicle is turning.” which proved to be way too big of a task. Perhaps there was a slight misunderstanding at the beginning of the sprints, but the team thought all tasks were *always* to be vertically sliced, which resulted in these big tasks. This is however not the case[1].

Besides the sizes of the tasks, the *definitions of done* (DoD) were consistently lacking. The DoD often prevented the team from actually adding value to the Product Owner by the end of the sprints (because the tasks were never considered done as they did not fulfill the criteria). Regrettably there were tasks without DoDs as well. This was problematic, because without a DoD, there was no way to know for sure when to move the task to done. Another reason to explain why the Scrum board was inactive is that the tasks were stuck in the testing phase. Many of the tasks were defined so that they could only be done if they were tested on functional hardware which lead to significant delays.

Another aspect that made the tasks seem less accessible was the team’s choice of estimation points. Instead of using an arbitrary point system to estimate the difficulty/effort of the task, the

[1] https://www.mountaingoatsoftware.com/blog/the-difference-between-a-story-and-a-task

team chose to estimate in time. That became confusing very quickly. For example, a task could have received “8 hours” as the given effort, but in reality, it took 2 whole days to finish the task. By having a confusing point system, it affected many other areas negatively as well such as the burndown chart (see more at KPI/Estimation Accuracy). As written, the burndown chart was not updated on time, and this is partly due to the team members not knowing if they should have filled in their actual working hours or the finished effort points.

2.4.6 Scrum of Scrums

To increase communication between teams, a weekly Scrum of Scrums meeting was conducted. In this meeting the Scrum masters of every group discussed what their respective teams were working on, as well as plans for future development and collaboration.

Using this resulted in synchronization with the other groups and gave insight on how far the other groups had managed to come. The insights also made the team branch outwards and seek help by those who succeeded past the technological bottlenecks. Besides getting insight on how far each group had come, common decisions were also taken regarding how we would collaborate for the platooning. All in all, this was very rewarding.

2.4.7 Project Communication Channel

For communication during sprints, a Slack team was used. This Slack-team allowed for questions and issues to be raised for everyone in the project to see, across all project-teams. It was also a means of reaching individual members or project-teams for collaboration and for organizing the sharing of the MOPEDs.

The team was very inactive in the beginning, and rarely checked what was being written on Slack. As the project progressed, the team realized the need to follow the conversations and get on board. Later on, the team got in more personal contact with group 1^2 (12 Moppepojkar), and as a result of this, the team obtained technological solutions which aided the team past the bottleneck(s), e.g. how to get custom code on the MOPED. Another example of this is when we had a non-functional MOPED for a whole week, and could not solve it since tech-support was unavailable. Fortunately, we had the insight to reach out and ask for help from other groups.

3 Guest Lectures

During the project, different companies held lectures on how they worked with software development. For instance they shared different problems they encountered as well as how they applied Scrum. This is very useful knowledge to learn from, and to put the team’s work in context with the real world.

3.1 Zenuity

Zenuity is a relatively new company that has been growing at an extremely fast rate. They develop software and systems in order to create autonomous vehicles. The lecture was relevant on multiple levels since our project, on a smaller and simpler scale, has a similar purpose. It was explained how, on a larger scale, their company and development teams were structured.

They emphasized the importance of agile development with continuous improvement and software releases as this results in frequent customer feedback which, in turn, provides a better structure for changes and continuous revenue. In our process, we felt that we lacked the frequent customer feedback but nonetheless we felt the benefits of continuous, small and incremental improvements. By delivering smaller increments, it was easier to see where we stood in relation to the vision of the product owner and also to adjust the product. It also meant that we did not lose much work if we had diverged slightly from the vision.

The importance of skill diversity was also brought up. This can be extremely important as problems that arises in bigger projects often are not one dimensional and often require skills from more than one area of expertise. In our group we lacked skill diversity somewhat. Particularly in the hardware department, and we felt the effects of that by being unable to solve the hardware problems on the MOPED. This severely hampered our development process as we were unable to deliver small increments in time due to our inability to test them.

3.2 IGDB

IGDB is a smaller start-up company. This lecture provided us with an insight in how a smaller company can run. The importance of experimenting with the different aspects of Scrum to see what works best for your specific group and project was particularly stressed. The importance of early delivery, and specific roles was also emphasized as it gives a certain level of responsibility and control, which can increase efficiency.

By being a smaller company, about the size of our group, this lecture was more relevant to our own process. In our group, it was hard to follow Scrum properly as we did not have a dedicated Scrum master for the whole project. Like IGDB, we could not afford to have a full-time Scrum master due to the fact that this would impact the development process negatively. They did however follow Scrum actively at all times and noted the importance of this. We noticed the consequences of not doing the same as we deviated a bit from some parts of Scrum in the end of the last sprint.

3.3 Volvo Cars

The guest lecture given by Volvo Cars showed the importance of having best practices. Since Volvo is a big company with thousands of employees, it is crucial to have a standardised way of

working. Their best practices did not come out falling out of the sky: it was something they've worked hard and long on to get to, and still something they're trying to improve.

One of their best practices was how they used Configuration Management (CM). A part of CM is version control, and they had a certain process to be consequent and uniform with their version numbers etc.

In comparison, the team has also had to develop certain practices to get working, but with considerably less time to really establish best practices. It is fair to believe that the work flow was negatively affected by the suboptimal practices.

As for version control, the team did not establish any certain framework for git usage. The only rule that the team introduced was that the team member should always pull down the latest version before pushing new content, in order to avoid merge conflicts. This was to relieve the work flow.

To round it up, the team has learnt and understood the weight of having well developed practices, because despite being just 10 people, which is a very small number in relation to Volvo's employee number: poor practices lead to big problems.

3.4 General Conclusions

The lectures mainly focused on how their companies worked and how they had implemented an agile framework. While this said a lot about the importance of learning and implementing these frameworks ourselves, what actually helped us for this project were smaller things within these massive strategies. These things include the importance of groups with specific tasks, and the responsibility of doing what you are assigned. The lack of skill diversity was also apparent as this project simulated a real project very realistically, where different skill sets could have helped immensely. Problems or issues more closely related to hardware or server were relatively hard for us as a group to deal with, and this strongly dictated our decisions when it came to our strategies for development.

4 Reflection on the Sprint Reviews

The meetings with the product owner were supposed to take place on thursdays. However he was only present two times, therefore our sprint reviews relating to the product owner are slightly limited.

During the first sprint review we had not made any proper increments other than driving the MOPED with the app. Therefore there was not much to ask other than specific questions about the vision, including what the focus should be. This helped us prioritize our work in later sprints.

In the following sprint review, there was not much to show as few increments were made due to various problems that had to be solved. But, it did help us to refocus and reorganize the group

such as moving more individuals to work with the ACC and learn python, by identifying where we stood in relation to where we should be. Therefore it was important even though we did not get to meet the product owner often. They were extremely useful at directing the focus of the group.

Optimally we should have had things to show to the Product Owner and ask him about from the very first sprint review, and more than two meetings.

The infrequent meetings with the Product Owner were because of external factors, and not something anyone could have done anything about. The problem with not having enough to ask or show to the Product Owner could have been solved if we had communicated more with the other groups, and asked for more help.

5 Reflection on the Relationship Between Prototype, Process and Stakeholder Value

Initially, the scope was quite big in relation to the end result and prototype. The stakeholder wanted as much functionality as possible and no one really knew what was realistic. The scope got smaller and more manageable as time passed by and as we held discussions with the product owner.

The project's direction was, throughout the sprints, guided by these discussions with the product owner. As an example, when we realized that it was not realistic to achieve platooning, we asked the product owner what he felt had the greatest stakeholder value. The information we received was that the ACC should be the focus, and we therefore reorganized the group to satisfy the demands of the product owner. The prototype therefore had a direct relevance to the stakeholder value, and the stakeholder value affected the way we worked.

6 Reflection on the Sprint Retrospectives

The sprint retrospectives were largely satisfying. They identified many issues with our process and defined strategies for resolving them. By following these strategies the workflow improved over time.

The main issues with our retrospectives have been:

- Not all problems were identified during the retrospectives. As an example, early sprint retrospectives had a generally positive view of the team's organization and workflow, and failed to identify problems such as the lack of adherence to the social contract. Our low truck factor was not identified as a problem until after it had caused a minor accident with the battery charger.

- The strategies defined during the retrospectives were not always followed**.** For instance, the lack of daily Scrum meetings was repeatedly brought up in the retrospectives, along with suggestions on how to make sure that they would be held daily. Yet, these strategies were rarely followed, and a lack of daily Scrum meetings continued being a problem throughout the project.

Ideally, the retrospectives would identify all problems and potential problems that could affect the project. In practice, this is likely not possible, as there are bound to be a few issues that manage to go unnoticed by even the most meticulous observer, but it is nevertheless a good ideal to strive for. A method for noticing more potential problems could have been to take a more proactive approach during the retrospective and ask what could cause problems in the future in addition to asking what is causing problems at the current moment.

The strategies defined during the sprint retrospectives ought to at least be attempted. The main reasons why some were not is because they either turned out not to be doable in practice or that we suspected that they would not be effective. The first reason is a legitimate one, but being afraid that a proposed strategy would not be effective is not sufficient enough to continue with one that demonstrably isn't.

7 Evaluation of D1 - D4

Listed here is the evaluation over D1 to D4; what we thought was right and what actually is.

7.1. D1 - The Lego Exercise Review

The Lego exercise and evaluation was a positive experience and an eye opener to some extent. We realized how important structure, planning, and having an incremental strategy, e.g. Scrum, are in order to provide results in a group project.

We decided on three strategies after the exercise.

- Have daily Scrum meetings
- Use Trello as a Scrum board
- Have sprint retrospectives complemented with "Happiness surveys" at the end of the sprints

As these are discussed throughout the report they will mostly be analyzed in perspective to what we expected of the strategies at the time of D1.

Our goal was to have daily Scrum meetings to make sure that everyone knows what is being worked on and the status of the sprint, to help bring up problems quicker and receive help if necessary.

We decided to use Trello as our Scrum board which was supposed to, together with the daily Scrum meetings, make the sprint easier to follow and increase the availability of the present tasks. These strategies were also meant to make it easier to split the tasks between members, and at the same time keep sharing knowledge within the group.

The development flow was simply non-existent during parts and periods of the project as we constantly ran into problems, such as hardware issues, that required outside help. This led to our day to day progress often lacking which affected our Scrum usage and increments. At times we skipped daily Scrum meetings because no work had been done and there was nothing to discuss. In hindsight, having the daily meetings to go above and beyond to fix these issues, or getting help from someone that could, would most likely have helped us advance faster in at least a few cases. With a better flow we would also have been able to set up more tasks which would have increased Trello activity as well.

With sprint retrospectives and happiness surveys we could see how the sprint had gone and how it affected our mentality towards the project and group. The original idea was to have daily questions, in connection with our daily meetings, regarding happiness to quickly bring up and fix issues as quickly as possible. The discussion was however only held during the retrospectives. As the group considered all results from the surveys to be acceptable, nothing more was done regarding these. With more time and more regulations the Happiness surveys could have had more impact on the project, as everyone agreed on the importance of it. Still the knowledge on how to act accordingly was not enough.

We also discussed how and why we selected our specific KPIs and how they were supposed to help us track the progress. This is analyzed in the KPI-section of the report.

7.2. D2 - The Initial Product Backlog and Vision

In the first backlog, we added anything that we could come up with relating to the MOPED and platooning. The reason for this was to create a diverse base that would allow us to prioritize later. This also led to that several user stories and epics remaining untouched throughout the project. The initial backlog was defined before the lecture on how to slice the user stories vertically, and they therefore were mostly horizontal. To adjust them, most of the user stories had to be rewritten.

In the vision, we discuss wanting to have a safe system, although we wrote that is was not a priority. At the end of the project, we reprioritized and made safety the number one focus. This means that we deviated slightly from the vision. As such, safety became the one thing we managed to achieve with our ACC. The reason for the change in focus was to avoid crashing with the MOPED. We realized that the MOPED would malfunction after a crash and therefore crashing made it harder to later test new features on the MOPED. In addition to that, we came to the conclusion that if the system was to be implemented in reality, safety would also be the first focus.

In the vision, we also specified innovation as a cornerstone. In the ACC, we felt that the project did not allow for much innovation, and innovation was useful first in achieving lateral control. For the lateral control we had two different algorithms that could identify either an edge or a bar code, meaning it had to be attached to the MOPED in front of our own. In the vision we wrote that our solution should work independently from the adjustments to the other MOPEDs, but we quickly realized that was not feasible within the 5 weeks we had.

7.3. D3 - Halftime Evaluation

Up until the half time evaluation we had not gotten much done. At first we didn't have access to a MOPED, and this was followed by hardware issues preventing us from getting started right away. We had gotten a wake up call the previous week during a lecture regarding actually following strategies that were presented to us. And now we had acceptable tasks that followed the INVEST criterias, albeit a bit too big sometimes. Even if they did follow INVEST, they were not precise enough for development to be made each and every day. In the beginning, a lot of tasks were not clear enough, and were pushed along, and not completed within a reasonable time frame as they lacked a distinct goal and definition of done.

We recognized the passivity issue that we suffered from, that often lead to us sitting around not doing much when we encountered problems outside our comfort and knowledge zone. This was improved to some extent after the halftime evaluation as we got a better understanding of the platform and more actively tried to resolve problems, such as hardware issues. For the group to have gotten further with the project it would have needed even more responsibility and more initiative taken by the members to solve problems as soon as possible.

7.4. D4 - The Working Prototype

Until the last two weeks of the project, we still had hope of achieving platooning and a safe ACC. Unfortunately, we were unable to achieve platooning, and thus not fulfilling the product owner's vision completely, but we did have a working app and ACC. However, we were unable to demonstrate every working aspect of our ACC due to integration problems with the app and the ACC. We were, however, still able to demonstrate a slightly more limited ACC. The feeling among the group is that we could have delivered a much better prototype in the end. We certainly put in enough hours, but communication errors resulted in a lot of things being implemented too late and therefore not ending up working as intended. The CAN-system of the MOPED that we had been working on inexplicably stopped working just before the demonstration, however this is not an excuse as this could have been prevented by testing our code on other MOPEDs beforehand.

8 Sources

[1] https://www.mountaingoatsoftware.com/blog/the-difference-between-a-story-and-a-task

[2] https://github.com/felixnorden/moppepojkar

[3] https://github.com/hburden/DAT255/blob/master/Slides/L2-Kata.pdf

[4] https://github.com/hburden/DAT255/blob/master/Slides/L9-Zenuity.pdf

[5] https://github.com/hburden/DAT255/blob/master/Slides/L10-IGDB.pdf