Reflection report

Group: Byggare Bob

Linus Berglund[1], Johannes Edenholm[1], Hugo Frost[1], Hamza Kadric[1], Erik Källberg[1], Johan Svennungsson[2], Rikard Teodorsson[2], Timmy Truong[1], Arvid Wiklund[1] and Karl Ängermark[1]

[1] *DAT255 – Software Engineering Project, Bachelor of Science programme in Computer Engineering, Chalmers University of Technology*
[2] *DAT255 – Software Engineering Project, Master of Science programme in Software Engineering, Chalmers University of Technology*

October 27, 2017

Contents

1 Introduction

This report covers the progress and result of our project using the SCRUM methodology by implementing a semi autonomous MOPED-car (Mobile Open Platform for Experimental Development) to accomplish the task at hand. Here we elaborate on how we used SCRUM, what problems emerged, how we solved them and making a retrospective analysis one the project. We have been given the oppurtunity to discover and expand on the problems that engineers all over the world face each and every day. Additionally, the theme of the project has been autonomous driving which is a new sprouting area within software development.

2 Application of Scrum

2.1 Roles, team work and social contract

As for the roles, the team only held a formal vote for who was going to be the official *Scrum Master* (Hugo Frost) of the group. Additional roles later emerged i.e. *vice Scrum Master* (Karl Ängermark), *Secretary* (Linus Berglund), *vice Secretary* (Timmy Truong), however none of these were official. The roles themselves were not decided nor carefully planned, they appeared as the group needed.

The Scrum Master acted as the group's outer face and messenger, making it easier to communicate with the other groups and their Scrum Masters. Nevertheless was this not the only task for the Scrum Master. The tasks would vary from delivering the MOPED-car to another group early in the morning, to collect information from the other groups as well as sharing it. To sum up, playing the Scrum Master role took up a rather immense amount of time, as the role could have been depended on anytime of the day, even after study time.

At times where our Scrum Master could not attend important meetings such as *Scrum of Scrums*, we had our vice Scrum Master to participate. Although it was only on very few occasions that our Scrum Master was not able to participate, having a so-called *"back up Scrum Master"* greatly helped maintaining order in our group.

Our secretary's role was to maintain effective records and administrate our meetings and assignments. It happened a couple of times that our primary secretary had other arrents to do, but this did not affect the workflow because the vice secretary could take over the role and handled it well.

Our social contract, linked here, helped to keep the morale high and have respect for one another. Everyone was aware when they did a mistake or broke a rule established in the social contract. For example being late to a meeting or not telling that a task will not be completed in time. This lead to a great

understanding of each other and no heated arguments erupted.

2.2 Used practices

During the project we have used different practices to improve our workflow.

2.2.1 Stand-up meetings

During one of our first meetings we went over the team's combined schedule with all courses and decided to have two stand-up meetings per week; each Monday 11:45 right after the scheduled lecture in DAT255 at Chalmers Johanneberg and each Thursday right before the supervision in Open Arena at Lindholmen Science Park. At this point of time we didn't realise that some of DAT255's lectures would only be placeholders to allocate work time and not actual lectures.

We felt that our stand-up meeting on Mondays rarely gave anything of value. Fridays were the first day of the next sprint and we weren't supposed to work extensively on weekends, so most team members had rarely done something of value and therefore did not have much to report on Mondays. It was however more valuable in terms of getting everyone together and deciding when to meet next time.

2.2.2 Work meetings

Thursday afternoons in Open Arena became our meetings for planning the next sprint and reflecting about the previous one. After meeting with the product owner and demoing our accomplishments, scrum supervision and the scrum of scrums-meeting we had everything we needed to plan the next sprint. This was done by using a whiteboard where we listed different potential user stories and then assigned at least one member to each listed user story, often multiple assigned members.

2.2.3 Subteams

Since a team of ten to twelve persons are slightly larger than the "ideal" Scrum team size, we decided early that one of our strategies would be dividing ourselves into subteams when necessary. Rather than completing tasks or user stories individually, working in subteams would allow smaller teams to work on similar problems and be able to help each other.

The strategy was mostly used in the beginning of the project, e.g. one subteam was focused on getting the server up and running while another subteam tried implementing some plugins. Subteams came more natural the further we came into the project, often when we realised some user stories were too big for one person.

"Pair programming" could have been a solution for enforcing our strategy harder, subteams come even more natural when in pairs. We had one situation quite far into the project when two team members were working on the exact same image recognition solution, essentially making the same work twice. Eventually both had to sit down and merge their two solutions into one.

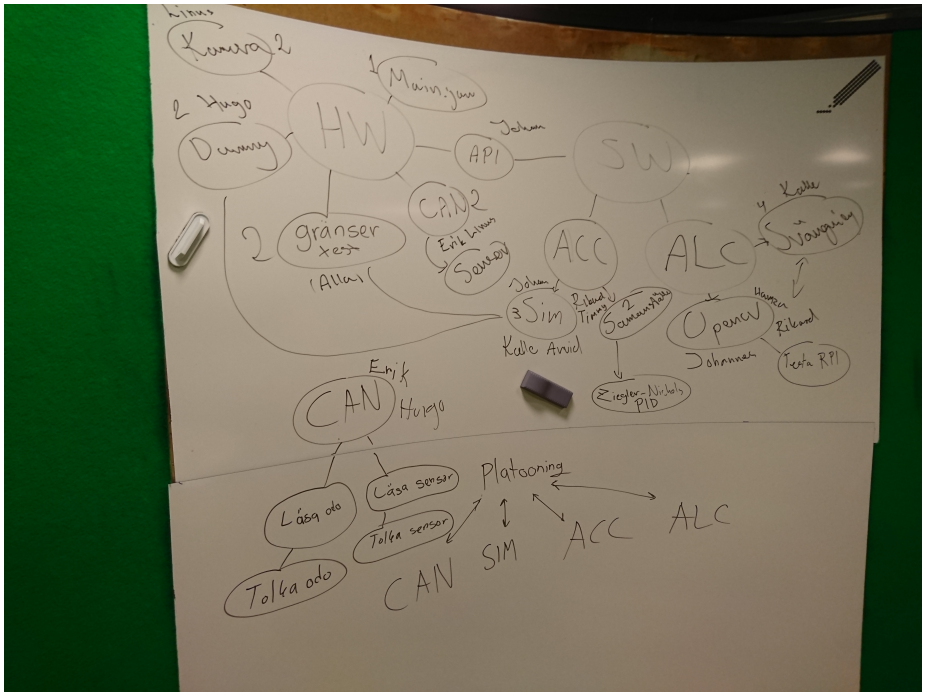


Figure 1: An example of our planning before a sprint

2.3 Effort, velocity and time distribution

The team followed a score scale of 1 to 5 regarding time distribution for every user story. Whereas 1 meant the user story would take a maximum of one hour to complete, whilst 5 indicated that it would take the whole sprint. On very rare occasions we would also put a time estimation score of 6, suggesting that the user story would take more than the whole sprint to finish. As aforementioned, the team ensured that the user stories were divided by having at least one assignee on each user story. Our team also preferred that every team member assigned themselves at least 5 points worth of user stories in terms of time estimation.

Furthermore, people were instructed to invest around 20 hours of active work on the project every week. Our team had, as aforementioned, *stand-up meetings* and *work meetings* to lay out during the sprints. Whereof an estimation of 8 to 12 hours per week were invested in attending meetings and working together

for longer sessions. The remaining hours were considered more as free working hours for whenever each person of the team had time. How well our user stories were defined and distributed would also affect on how much time each person contributed. Meeting up with the group, attending lectures and all the working hours included could from time to time not truly be measured up to 20 hours per week. Therefore it would sometimes feel like insufficient amount of time was invested during certain sprints.

On the contrary, the team would always be updated about the other teams after each sprint, which would reflect on whether all the groups were on the same page or not. This would remind us that the very importance is not what has been accomplished among the other groups relatively to our group, but whether our group managed to carry out what was needed to be done in our *sprint backlog*.

Our approach to distribute time during the sprints would on occasions face a few difficulties. The team would agree on meeting a specific number of times during the sprints, however having *meetings* were not by definition getting work done. Having too many formal meetings (featuring approval of agenda and consistent updates from subteams, although barely one or two days passed) was too time consuming. Therefore the team came up with the solution to convert some of our formal meetings or *stand-up meetings* to plain *work meetings*, where the group solely needed to focus on getting productive work done.

At particular sprints where most of the user stories were completed and the working hours for the week still not fulfilled, our group realised that it was not a matter of how many hours we spent on the project itself every week. So even if we had meetings regularly during our sprints and the objective time was roughly fulfilled, it did not imply however that most of our work were completed.
In other words, the true meaning was quality over quantity regarding the time distribution.

2.4 Task breakdown

Our user stories could have been broken into smaller tasks, which would have made it easier to know how much was left until it was completed. Many of them were horizontal instead of vertical and didn't deliver anything of value to the product owner. They were instead important to the team.

3 Reflection on sprint retrospectives

Our sprint retrospectives were held during our work meetings on Thursdays, at the end of our sprints. We felt this was a good time to reflect on the past week, as everything was still fresh in the mind and we would be able to voice our concerns before planning the next sprint.

Looking back on the retrospectives, it was quite difficult to make the discussion

stay on the topic of the general workflow and our process. Many times, it would veer towards specific technical challenges and semantics that could be discussed at another time. This is indicative of how important and challenging it is to take a step back and take a look at how the project is progressing and what the next step should be.

An interesting thing to note, looking through documentation of the retrospectives, is that they mention Scrum-specific things (user stories, assignments, stand-up meetings) less as we progressed. For example, mentions of user stories in the last three sprints just states that we were happy with them. The new problems we faced did not arise from inept application of Scrum but were actual technical problems. This is one indication that we managed to learn how to create good User Stories and plan our process better as we went along.

The retrospectives did not have any particular structure. Before planning the next sprint we would take some time and have an open discussion. This is reflected in the fact that the documents we wrote differ quite a lot in length, format, language and content. A different approach could have been to have a list of bullet points set from the start that would be discussed each week. First we would discuss our user stories, then problems we have, then ideas for the next sprint etc. This might have made the overall progress we made in applying Scrum more apparent.

3.1 User Stories

Our understanding of the User Stories changed a lot during the course of the project. We knew the importance of creating vertical tasks from the start but actually applying it was harder than suspected. Our first week we gave everyone the same User Stories (researching the documentation, setting up git etc.) and also one which was uploading code to the RPi, which was something we did together. On the one hand everyone could complete this on their own but the definition of done was quite diffuse and the time each team member spent varied a lot. The issue in the early weeks was time. It took a lot of time to come up with tasks for everyone to do that would amount to approximately 20 hours (including meetings). Additionally, discussions would often get off track. We managed to speed things up by being more focused by having people in different sub-teams. Each sub team would be working with different areas of the project (ALC, ACC, Simulation etc.) and basing our task distribution on that made things easier.

4 Reflection on sprint reviews

At the initial sprint review we did not have anything to demo since our first sprint tasks had merely been to set up a development environment for ourselves and produce a commit message to our git. A task of which all of us had

accomplished. Our review of this task was simply "Great job!".

After the first sprint we were informed that several groups have had difficulties with the server, and therefore reformed to a Python based solution instead of Java.

By the second week we had had a few complications with the server, which led to us not being able to attain all our sprint goals for the week, thus did not have anything to demo and came to the realisation that we might need an alternate solution. By this period, the group also had difficulties starting out with developing our ACC since no information could be obtained from the sensor, we were suggested to continue our development and in the need of data, direct fake data until further notice.

After the third week we felt something could be sampled, since we decided not to apply the server solution for our plugins. This week we could show that data could be sent to the MOPED-car to remotely steer and "start" the motor via a laptop. The review from the product owner this week was a 6.5 in a satisfaction scale from 1 to 10 where 1 indicates that we have no chance of achieving the final demo, and where 10 meant that we would. We felt optimistic with this review, although we felt we still desired to work fiercer to carry out our goals.

For the fourth week we succeeded to demo an effective ACC, and the product owner stayed pleased with us by the terms that the vehicle could drive in a straight line after another vehicle and stop if the front vehicle stopped. Our evaluation for this week was a 10 out of 10. As of this we felt we now had accomplished our task and did not feel as motivated to further develop our solution, primarily since numerous MOPEDs were out of function. Although, we still sought to implement an ALC and agreed to hereby modify nearly all our meetings to work-meetings, where instead of conversing, worked on our MOPED, to advance it. This being the first week we felt we had accomplished something.

During the fifth week we enhanced our ACC remarkably, making it stable and specific, since it now ran in 40Hz. Showing this in our demo, we once again obtained the grade 10 out of 10 on our satisfaction scale. By this time our ALC was partly working, however, we had not implemented it in our MOPED yet.

After our final presentation the product owner remained content with our presentation and encouraged us to feel the same way, although this was arduous for us since we did not receive a "faultless" MOPED to evaluate our solution on. The vehicle that was fetched had a steering difficulty and its natural forward setting was crooked, slightly to the right, which made our lateral control (ALC) have issues following other vehicles when they steered left. It made us feel disappointed that we could not view our MOPED in action, and we were sure that we could follow another vehicle if we had a fully functioning MOPED since it performed closely to flawlessly when steering right and following another vehicle in a straight line.

5 Best practices for using new tools and technologies

5.1 Scrum board workflow

Github has a built-in feature for project boards, which we used extensively during our sprints. We created the following six columns from left to right.

- Product backlog
- Sprint backlog
- In progress
- Testing
- Done
- Abandoned

The workflow for creating a user story was to add a Note to the Product backlog-column which creates a "card" and then converting it to an Issue, which is another built-in feature by GitHub. In the Issue's description-field we defined Definition of Done and the User Story itself. Once that was completed, we added Labels to each user story.

Example of Labels used:

ACC short for Adaptive Cruise Control	**plugin** user stories related to creating plugins to be deployed on the MOPED.
ALC short for Adaptive Latitudinal Control	**server** user stories related to the getting the server-solution up and running.
environment user stories for setting up your favorite IDE, `git` etc.	**S1,S2,...,S6** the sprint when the user story was moved from Product backlog to Sprint backlog.
high priority crucial user stories that were to be completed first if team members were assigned to multiple issues.	**SW implementation** software
HW communication hardware	**technical debt**
low priority	**theme** large user stories that would require multiple user stories to fully complete.
medium priority	

5.2 Git workflow

The group chose to adopt most of the model *A successful Git branching model* by Vincent Driessen[1]. We used `master` and `develop` as our main branches.

`master` was decided to be a "protected" branch, which means only the owner (Erik Källberg) of the repository could merge other branches with `master`.

We did not use any other supporting branches other than `feature` branches, which were branched off from `develop`. After each sprint review the goal was to merge all `feature` branches into develop and then merge `develop` into `master`.

During one of our sprints we had some complications with the workflow. `feature/X` was not finished and had not been merged with `develop` during the previous sprint. It was therefore several commits behind `develop`. Code that had been implemented and pushed to `develop` was needed to finish `feature/X`. So instead of merging `feature/X` with `develop` and creating a new remote `feature/Y` branch, we decided to merge `develop` into `feature/X` instead. We learned that `git rebase` can be very dangerous command while merging. The commit history was revised, commits done several days ago was added to present on `feature/X`. The incident was resolved by creating a new branch from a commit before the merge.

Even though we adopted a recommended workflow, we did not really use it as intended throughout the project. Feature-branches were only merged with `develop` *after* each sprint, so in other words; we used `develop` as another `master`-branch. If we had merged finished feature-branches more often, forbidden merging feature-branches with each other and had a more "relaxed" view on `develop` in general some minor git complications could have been prevented.

5.3 Slack

Since there were a Slack channel for the course to ask questions and get information it felt natural to create a channel for our project group as well. It helped us immensely to solve logistical problems and organise meetings. It also made it a lot easier to follow our social contract which stated everyone had to be called before meeting time, later we realised this needed to be done one day before to not cause inconvenience. The feature to get announcements from GitHub was also greatly appreciated since this meant one could track the other subteams progress and notice any problems in the git repository as early as possible.

5.4 Choice of programming language

We decided to make all programming in Java since that's the language everyone felt most comfortable. The only risk noticed going with Java was if we could not rewrite the Python files to Java. This would have meant a huge problem if it had occurred. Luckily this did not occur and Java worked wonderfully for this project.

5.5 CAN communication

After our third sprint we gave up trying to get the existing server solution with plugins working. One other group seemed to be making decent progress by communicating directly with the MOPED by using existing Python-scripts and bypassing the server solution entirely. Since only two out of ten team members felt somewhat comfortable writing in Python and the "truck factor" would be too high, we settled for a slightly different approach. After code reviewing the Python-scripts we noticed that they were communicating directly with the CAN bus, so our hypothesis was that we could achieve similar effect using Java code instead, which all team members felt more comfortable with.

The subsequent sprint demo with the product owner we had implemented a CAN-interface for communicating with the MOPED using code written in Java. This essentially means that we in one sprint made the existing server solution with plugins obsolete.

5.6 PID-regulator

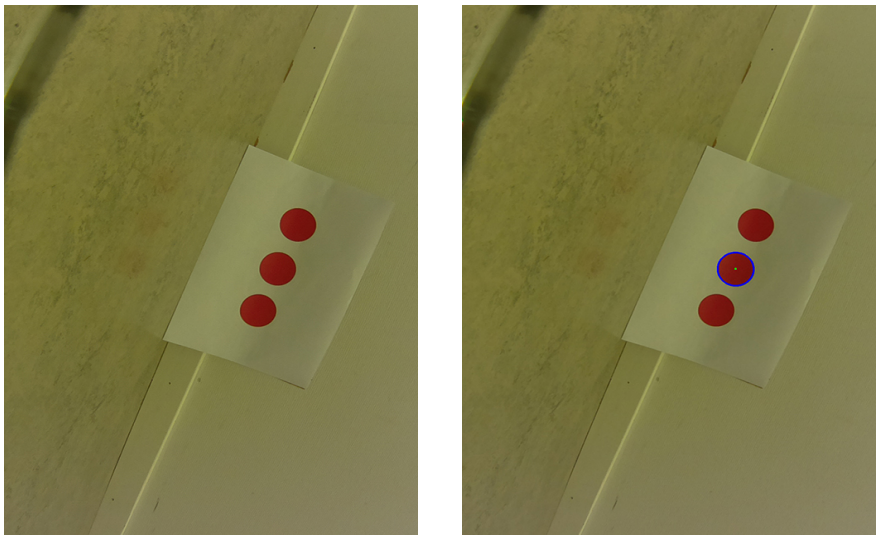
To complete the project there had to be functional ACC and ALC. Since a majority of the group had taken the course "LEU236 - Dynamical systems and control engineering" we decided to apply the skills acquired from that course and construct two different regulators. The only alternative approach we saw was to make some sort of state machine. For example setting a fixed speed depending on sensor values from the MOPED. Although this approach would have been an easier initial implementation, it would not have scaled well and it felt unresponsive and unwieldy.

A PID-regulator uses three constants to determine how strongly each part should affect the end result. P is the amplification. I is the integral part, adding the the error received and letting it accumulate overtime. D is the derivating part, comparing the last error with the new value and letting the difference affect the regulator. The result is equal to the sum of the different parts.

To go about using a PID-regulator some of the members got a user story to look up how it could be implemented. This included reading literature from the LEU236 course and checking general implementations in programming. This meant we had a skeleton to base our regulator on and could move on using tests to evaluate the different constants needed.

5.7 OpenCV

Searching for an appropriate image recognition technique, we gave three group members the task to simply look up on the Internet for a way to detect shapes/colours or both on an image with Java. When this had been done we met and discussed our alternatives. All three members of the group had found OpenCV to be the



(a) Input before OpenCV (b) Output after OpenCV

Figure 2: How OpenCV detects the centre circle

best choice for this task since it already had a lot of predefined methods for finding both shapes and colours in an image[2].

This lead to two members working separately on an image recognition solution for finding a red circle on a piece of paper. Both members managed to get functioning code for this, but with slightly different techniques. One used three red circles for better accuracy and the ability to filter out, if the camera would snap up other red objects in the shot, whilst the other instead had a solution on how to read in different pieces of images correctly depending on the light in the room. After working separately and then showing and explaining their solutions for one another they simply decided to merge together their solutions and take one members code on how to find three circles on a piece of paper, and the other students technique on how to read in the image and finding the red in this image.

6 Relationship between prototype, process and stakeholder value

After finishing the sprint backlog for the upcoming sprint we also dedicated time to discuss what could potentially be shown or demonstrated during the next meeting with the product owner. This way we all had a common ”sprint goal” in our mind on which user stories should be prioritised for realising the prototype. By always having something to show the product owner at the end

of each sprint we were positive we had produced *something* of value, but not necessarily stakeholder value. By comparing what we were able to demonstrate the previous sprint with the current one we also received confirmation that the project was making progress.

Since one of our key performance indicators (KPI) was directly related to measuring stakeholder value – asking how satisfied the product owner was with our sprint efforts based on the demonstration – a higher score from the product owner would indicate that our working prototype and the produced value was in fact stakeholder value.

7 Our process related to literature and guest lectures

After hearing how companies like Zenuity, Volvo Cars and 8 Dudes in a Garage are applying Scrum in their work environment we all consider the Software Engineering Project to be a very relevant course if we were to set foot on the job market tomorrow.

Zenuity specifically told us about newly recruited senior engineers from Volvo Cars that have little to no experience of working according to Scrum before. "Teaching an old dog how to sit" might prove challenging for Zenuity since they don't have eight weeks of beginners' training in learning Scrum. By the time we graduate, we should have a more advanced level of understanding of Scrum.

8 Evaluation of D1-D4 assignments

During the course we have had the opportunity to stay focused and progressively evaluate our progress with the help of reflective assignments. D1 was our first requested assignment in which we reflected on how the group had performed during the Lego Scrum exercise and how our approach to it could have been improved. We did find the exercise to be very useful because we quickly realised the strengths and weaknesses of the group. Trying to construct a product without splitting up the work and delegate specific tasks to each member proved difficult and slow-paced. In the end we started delegating instructions which led to much quicker progress. Looking back at this we had found our first strategy for the MOPED project, that is, delegate work to sub teams. Our KPI for this strategy was to give points between 0 to 100 where 0 points represented a completely failed sprint while 100 points meant full completion of assigned tasks for each sub team.

In the D2 assignment we delivered a vision and our first backlog for the project. In the following lecture our backlog and its user stories got anonymously criticised by the teachers for not following the vertical and horizontal model of

dividing the work and also having bad descriptions. After this event we tried to follow the guidelines and the team unanimously share the view that we have been able fulfill this. In hindsight, the group have realised the importance and usefulness of decomposing the work into smaller slices, i.e. making a carpaccio of our user stories and try to apply the INVEST logic on them. This resulted in a switch from the usual waterfall curve to constant value growth after every sprint ending, which in turn gave better feedback from the teachers, product owner and within the group. The second part of the D2 assignment involved specifying a vision for the group's system. Even though it resulted in just a small piece of text, we think it is a crucial part of the project. There were certainly changes in features and methods but the vision remained the same during the whole course. Without sharing this common idea it would have been harder to communicate and individually perform something everyone was satisfied with. Thanks to the product vision, the goal became clearer and clearer after every sprint.

In our half-time evaluation (D3) we described what had been accomplished so far and what kind of remaining issues we needed to solve. In retrospect, we can say that we had a good understanding of Scrum and had implemented the method very well within our framework. It helped us diversify, divide and plan the project better than in past courses. Now with the practical work being finished the group still share this opinion. We also observed that the weekly sprint periods were adequate for this project. The group did still experience imprecise time estimations. However, we think that is unavoidable when using new unaccustomed tools and methods.

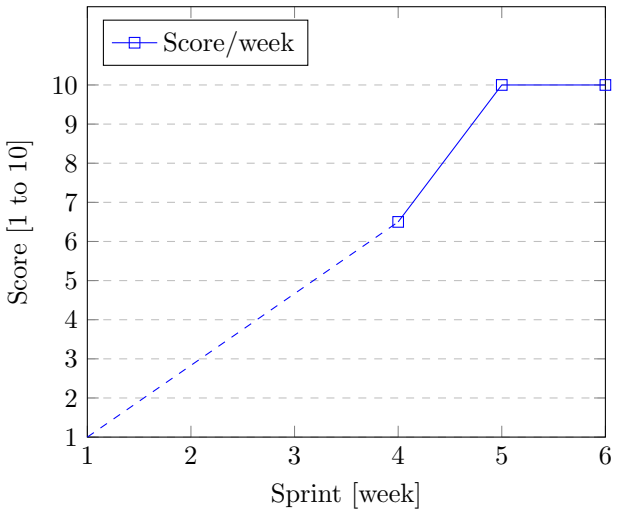
Looking back at the half-time review we see the mixed results from our try to implement a server solution. Even though we strived having smaller slices, we were not able to create a server for the MOPED. Mainly because of lack of basic knowledge, which needed to be obtained in a brief amount of time, thus we decided to abandon the idea of a server solution. Rome was not built in a day, so had we had a couple of more sprints perhaps we would have been more successful. The group struggled to uphold Scrum practices in the last two sprints due to stressfully prioritising delivering a working prototype for demonstration. Being pressured and still use planning methods is a positive sign of professionalism. Finally, having meetings where everybody can express their opinions and reflect on the progress and the planned path to reach the vision is an excellent way to estimate the development and show appreciation.

Our final demonstration was a stressful day. We decided to meet early to try and implement our software on a new MOPED-car. However, the day turned out to be full of hardware malfunctions as not a single one of the available vehicles were completely working. Not only did we try to solve the problems our MOPED had, but we were also asked for help from other groups which we gladly did. It took several hours until we had tested platooning with another group's vehicle. We had difficulties with achieving a fully working MOPED-car, which in our opinion led to a somewhat disappointing final presentation. Even

though we arrived very early it did not benefit us because we had to wait until the first combined presentation was over to acquire a working vehicle. However, the car still had a defect as the left front wheel could not steer thoroughly to the left. Despite this, our teacher Jan-Philipp Steghöfer still thought our presentation went better than expected and understood that most cars had defects. Therefore, the teacher thought we should be proud with what we had managed to accomplish, which made us feel that the presentation was still quite successful in the end.

9 KPI charts

Continuous feedback from the product owner



We only received feedback from the product owner in the last three sprints. The purpose of this KPI and the reasoning behind it was to easily compare the groups effort and general feeling of the sprint to the perceived value increment in the product from the product owners view. Furthermore we were able to get direct feedback on how well we understood the product owners vision.

After a couple of sprints we realised that the other KPI we chose were hard to quantify. Our second KPI was to delegate into sub-teams and rate their performance each sprint based on how many tasks and user stories were completed by each team. The third KPI was to divide the user stories to make them easier to handle. We did however reflect upon them at our stand up meetings and at our sprint reflections, going through each team, letting them tell the others what they have done and what was not completed. At the end of each sprint we divided our stories into smaller tasks, applying the reasoning behind the KPI in our work.

10 Conclusion

In previous project courses we have lacked the tools for a effective distribution of tasks and how to successfully prioritise them. This often caused shortage of time, unnecessary pressure and an end product that often could have been vastly improved if time and code planning had been done better. Now however, we were given this so needed tool called the Scrum model, with which we could much easier see what had to be completed, what was in progress and what had already been done. In addition, we now also had the ability to better divide the work based on priority.

Unlike previous project courses where a lot of time was put to do file merging in GitHub, in this course it was a lot easier and less time consuming to resolve them thanks to our Scrum-board on GitHub. We did of course have some inevitable minor issues regarding GitHub conflicts. However, a consequence of good breakdown and synchronised completion of user stories between and within subgroups made the whole process a lot easier and straightforward.

One conclusion that all group members agrees on is the importance of always having an overall plan and to be flexible of changing it when issues occurs. There will always emerge problems during the course of the project, so you should always be prepared to encounter this by having good code and time planning. Thus it is important to discuss both with external parties and within the group about the applicability of the ideas that is planned to be implemented. Otherwise, it will be unnecessarily complicated to reach the vision of the project.

Most team members liked the setup of lectures and supervisions as they guided us well to accomplish our vision with the project. The team commitment were at a high level and a friendly atmosphere prevailed. The lack of sufficient MOPED-cars and chargers for all groups and a bit ambiguous lecture scheme with unclear order of priority could be sorted and improved. It should be noted that we only see this as constructive critique for future editions of this course.

In addition to learning a new method for team cooperation and continue improving our programming knowledge, we have also realised the importance of having quality over quantity when studying our finished product. The key is to always deliver something of value during each sprint, because in the end you are creating the product after the product owners demands and not only for yourself. By using Scrum, we have seen that it is easier to receive feedback and deliver value to the product owners in each sprint. This course has positively enlightened us and shown why this framework for managing programming development is so popular in the software industry.

References

[1] V. Driessen. (2010). A successful git branching model, [Online]. Available: `http://nvie.com/posts/a-successful-git-branching-model` (visited on 09/06/2017).

[2] OpenCV. (2017). Opencv library, [Online]. Available: `https://opencv.org/` (visited on 10/02/2017).