CHALMERS
UNIVERSITY OF TECHNOLOGY

Agile Software Project Development:

A reflective report based on a small development team project conducted during the autumn of 2022

DAT257
Autumn 2022

Team Yavin:
Erik Hillestad Andreasson
Johan Birgersson
Lukas Lidvall
Viktor Rafstedt
Lukas Wigren

Introduction

Prior to the course even starting, Erik, Johan, Viktor, and Lukas Lidvall had decided to join the same project group since they were close friends from earlier. From this moment, the search for additional group members began, where Erik and Johan reached out to Lukas Wigren as they had previously worked on a project in another course. Lukas Wigren accepted, and thereafter the group Yavin was formed. Having a project group where every member at least knew a few others subsequently influenced how all members proceeded to engage in the course. Furthermore, it also helped a lot with finding a quick common ground in certain ideas- and design decisions. However, it also had its downside. It led to a relaxed mentality at the beginning of the course, which ushered us to build a sprint structure that was wrong, and, as we later understood, inappropriate due to the length and nature of the project.

As the course's start was right in the build-up to the national election in Sweden, we decided that focusing on furthering democracy as a sustainability criterion for the project was perfect. With a certain scope of ideas, we ultimately landed on an application form as a one-stop-shop for all relevant election- and polling information related to historic elections on a national, county, and municipality level. The idea was that this would serve as a great easy-accessible place to find any election-related data that you could possibly want to further democratize information about the Swedish democracy. In the initial stages of the project, we also developed a certain vision of a finished version with numerous extensions to what was first concretized, making the application a perfect stepping stone towards a much more extensive and important product, than we initially had realized.

1. Customer Value and Scope

1.1. Application scope and features

A: The scope of the project has been to create an application that displays Swedish election data in an efficient and meaningful way, initially the data was focused around election participation, but the scope early on extended to include more general election data as well as having robust options for future data expansions. The features mainly consist of being able to select an overarching data set to show, which in turn should be able (if relevant) to be graphed and manipulated through filtering or searching by the user. The stakeholder of the project was a computer science student at the University of Gothenburg and was seen as a potential consumer for the final product.

Workwise, we separated it into three main areas, backend, frontend, and data. Whilst there were some overlaps of who focused on what area, each team member had their designated area of responsibility which was adhered to throughout the project. There have, however, been some issues with the main areas of work since some areas were intensive at the beginning of the project, some areas were more intense late in the project as well as there was a natural difference in overall workload between the areas. This has led to team members going outside of their designated “work areas” to be able to contribute, which has led to some issues around clarity of who should do what.

B: We were overall satisfied with the scope, however, we were fairly late with including our stakeholder in the process, and it would have been beneficial to have had our stakeholder more involved earlier in the project when the scope was defined as some misunderstandings could have been avoided. Further, the same goes for the MVP. Whilst there was an MVP in place early on, the nitty-gritty details of it were defined fairly late, which led to some confusion about which tasks to prioritize within the group.

Lastly, regarding areas of responsibility, it would have been better if we had either put in more effort to divide the overall work required into better areas with a more even workload, or, if we would have had a system in place where a team member were responsible for the results in each area, but not over its execution, so that it would have been easier to have a fair separation between the weekly tasks.

A → B: Simply put, in order to achieve B the process of acquiring a stakeholder and involving the stakeholder in the process would need to hold a higher priority, especially early on in the project. Assuming that the stakeholder were more deeply involved early on, this would need to be leveraged in order to define a more crisp MVP. Furthermore, we would need to map out the expected work further into the future early on in the project, which probably would be aided by a more crisp MVP, so that we could divide up the work better.

1.2. Success criteria

A: Our success criteria can be separated into two main parts, where one is our own personal developments and the second is our deliverables. Regarding our personal development success criteria, we loosely discussed this as a group in the beginning but went into more depth on the topic in our individual reflections. As a group, we decided that we all wanted to extend our knowledge within Java, which was already fairly developed. We could have taken a different route and attempted something completely new, but we all agreed that extending our java knowledge was the way to go. The reasoning behind this decision was in part, that we did not want to focus that much on the “coding” aspect, but rather on the broader topic of software development which includes both software design and project management, which in this case has been agile. Thus, our success criteria in terms of personal development were partly extending our java skills, but mainly developing our skills in the agile framework and overall software design.

Regarding the deliverables, our main success criteria were to deliver on our MVP and to have an adequately satisfied stakeholder. We felt like we achieved this, but we realize that the application in its current state is not really usable due to a lack of data options, which results in there being several online tools that are in all ways better.

B: In regards to our personal development success criteria, we were overall satisfied with both the expectations we set up and how they were fulfilled, and would not have changed that much if we were to do it again. If we were to change something, it would probably have been in the area of software design, where we would have needed to study up on our software design skills more early on in the project since it is so crucial when it comes to design to get it right from the beginning and in some instances where we did not get it right from the beginning, which led to a somewhat chaotic design which we did not learn that much from.

On the other hand, in regard to our deliverables, there are some things that could have been improved upon. Firstly, in discussions with our stakeholder, we would likely try to define a scope that would have a more real-life use case (taking into account the natural limitations of the project). Hopefully, this would have led to even more clarity regarding what were the most important features as well as engaging the team more.

A → B: On the topic of personal development, there is not much to say here that has not already been said in B, however, it regards to our deliverables, the most important thing that we would need to do would be to involve our stakeholder both more and earlier in the project. Kind of in the same way as discussed in the previous section.

1.3. User stories and tasks

In order to keep track of our user stories, we made use of our scrum-board on GitHub (Yavin, 2022a). Our main focus early on when defining our user stories was to one make them mutually exclusive and collectively exhaustive, and second to break the tasks down with a top-down approach. We thought that we defined our user stories fairly well in the beginning and were satisfied with how we were able to keep them separated so that tasks could be worked on independently. However, in all honesty, we lost focus of our user stories as the project progressed, and whilst we adhered to the initial user stories, which were fairly exhaustive, we did not revisit them very frequently. This could have something to do with the fact that we initially misunderstood how user stories were to be used and, as we early on, had divided the project into two sprints. After this came up in the first supervision, we redefined many parts of our structure, including our user stories, which may have led to us losing track of the user stories from the overall perspective. However, the “purpose” of the user stories were not entirely lost to us as user stories were frequently discussed with our stakeholder, but we rather lacked in concretizing said stories into written user stories which we, in the end, could base our work upon. Consequently, this also led to acceptance criteria more often than not being related to tasks rather than user stories which incurred us with the risk of missing out on things important to the user to the benefit of things that were important to us as developers. All in all, it can be summarized that our task was based on user stories, but the written user stories progressively lost their use as we did not revisit them enough.

B: Obviously, there are some things that we ideally would like to do differently here next time. Starting with what we actually were satisfied with, the mutually exclusive and top-down approach would be something that we would apply again since, as previously mentioned, this allowed for efficient parallel working and a clear progression throughout the project. The first thing that we would have done differently would probably be to have a better scoring system for our user stories in terms of the effort needed to complete them, such as the Fibonacci sequence. Further, we would like to have our tasks more closely connected to a specific user story rather, as the way we had it mostly meant that the tasks were connected to the end goal and not a specific user story, which led to it sometimes being unclear exactly why you did what you were doing. Another thing that we would focus on is revisiting our user stories more frequently and adding user stories based on our stakeholder needs, as this would have probably made the user stories have a more clear meaning to us in the project, which would have motivated us to make more use of them, which based on what has been described would have been strongly beneficial. Lastly, we would have liked to have acceptance criteria more closely connected to our user stories. The main issue we perceived by having the acceptance criteria too closely connected with specific development tasks was that it could often lead to compromises that were not beneficial to the user but beneficial to us as developers, something that hopefully connected the acceptance criteria more closely with the user stories would have solved.

A → B: A lot of the mistakes we made in this area were due to a lack of knowledge early on in the project, so of course, knowing more about it would have helped. However, more relevant methods that would have helped us improve would perhaps have been to use a different tool for creating a scrum board, such as Trello, as we either did not find all the functionality available in the GitHub scrumboard or that it is not there, but it would have helped if we had more predefined areas that should be in user stories which we could have made use of.

1.4. Acceptance tests

A: As discussed in the previous section, as the project proceeded, our acceptance test were mostly connected to specific tasks that, in turn, was connected to a user story. We used the method where different branches were used based on different user stories, and tasks pushed to these branches needed to be reviewed by at least one team member (other than the one who wrote it) but preferably more than one team member. By the time it was deemed as beneficial to put the branch in its current state in the main branch (such as the GUI being needed to progress back-end tasks), and if it had been reviewed and accepted, the branch was pulled into the main branch. If there were no obvious interrelations between

the branch with other branches' needs, the final pull of the branch was made when the user story was deemed to be fulfilled.
There was no obvious pattern regarding which team member should review what. It was rather a more sporadic procedure. In the case of a team member not accepting completed tasks, it was either brought up in the weekly meeting or by direct communication between the team member who wrote the code and the team member who did not accept it.

Lastly, we made no use of Unity testing in our project as we did not feel that there was a real need for it other than for symbolic reasons.

B: One main thing that we would improve upon would be to have a more clear structure in who would review what. An issue that arose was that some team members became more used to handling pull requests etc., early on, which led to them doing a larger share of the accepting. This led to a somewhat skewed work distribution which was compensated for in other places, but it would have been beneficial if everyone would have been equally efficient in accepting tasks.

A → B: Based on the previous paragraphs, we would likely create a structure of either a turn-based order of accepting tasks or different team members being responsible for accepting tasks in different branches. Further, we would try to ensure early on that all team members are confident in git and especially pull requests so that lack of knowledge would not be something that complicated things.

1.5. Key Performance Indicators

A: Starting off in this project, we had not used any KPIs to measure the success of a project concurrently while working on the project itself. So we were slow to implement any KPIs in this course, partly explained by our inexperience in finding relevant KPIs that reveal the most information, while also being easy to measure each week. However, we landed on three KPIs during week 4. First, a burndown chart between the previous week and the current week with planned tasks versus actual tasks. This served as a production measure that we could look at and see how well we had planned and later executed the plans for that specific sprint. Second, a team satisfaction measure between 0 and 10, where each member of the team could rate their satisfaction with the other team members and the team's cooperation overall. When looking at this chart across many weeks, we can see that team satisfaction steadily increased over the course of this project. This is most likely explained by the common vision, the workflow within the team, and the knowledge of how to work together in an agile manner (including the scrumboard), that we formed along the way. The third and final key performance indicator that we utilized might be the most important one; the stakeholder score. As, in the real world, the customer is always right. If our stakeholder/customer is not happy with the weekly progress towards the planned delivery of their envisioned product, then there's a real problem. The score is chosen between 0 and 10 and signals how the stakeholder feels about the current value of the project for each measurement. This score, like the team satisfaction score, increased over time, and for the last measurement, our stakeholder scored an 8 out of 10, which left the team very happy. In hindsight, the KPIs truly was an important tool for the team to get on common ground about what we had done, the success of it, and how well we worked together. The alternative would surely have resulted in less of a common perception of all named measurements and thus weakened the effectiveness of the team. Therefore, this is probably something that we will apply in most future projects.

B: One thing that we could have made better, was to begin earlier in the project by involving a simulated stakeholder and measuring our productivity and member satisfaction. Mainly to get more data points to get an overview earlier of how the project is proceeding. Another important difference that we could have made is to perhaps include more KPIs that measure additional things or have broken some of the currently used KPIs into smaller KPIs to receive a more comprehensive understanding of what is not working well, and what we could improve upon. However, there is a fine line where too many KPIs might be distracting for the problems themselves, especially in a smaller project such as this. Therefore, one or two additional KPIs might have resulted in a more straightforward understanding of how the team is doing, and what could be improved.

A → B: The major obstacle from A to B was our inexperience regarding using concurrent KPIs in projects. With this in mind, for future projects, we would most likely initiate the discussion about tracking the performance of the project in the planning stages, rather than once the project has already commenced. With earlier discussions in future projects, more precise KPIs would be included, designed to cover specific areas, and at the same time, balance with not implementing redundant measurements.

Below are the graphs of our latest measurement for each KPI. Team satisfaction and stakeholder score show the progress across all weeks since the measurement began. The burndown chart, however, only shows the burndown of the latest week while working on the project, week 7.

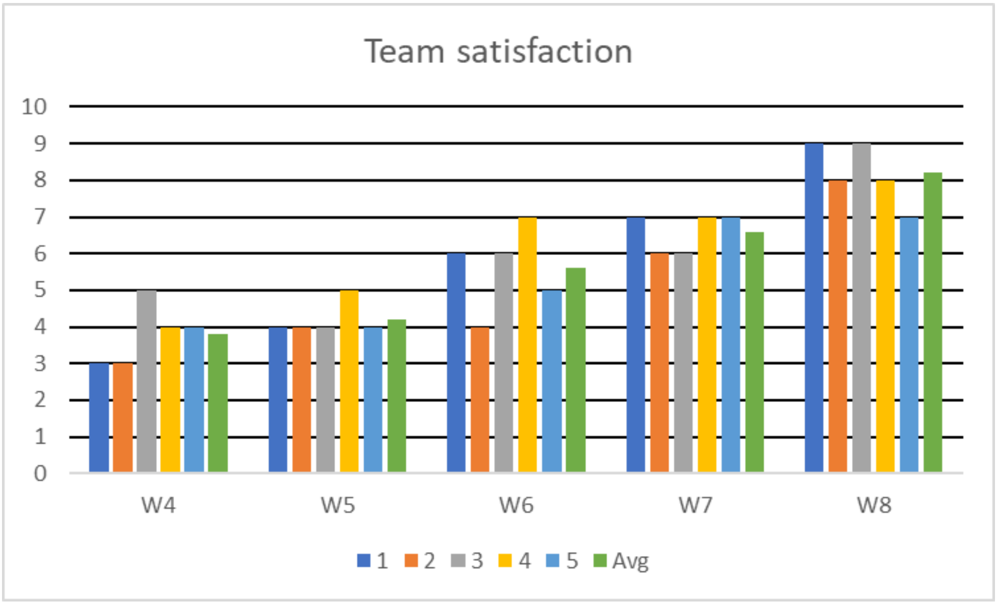


Figure 1: Team satisfaction between week 4 and week 8 of the project.

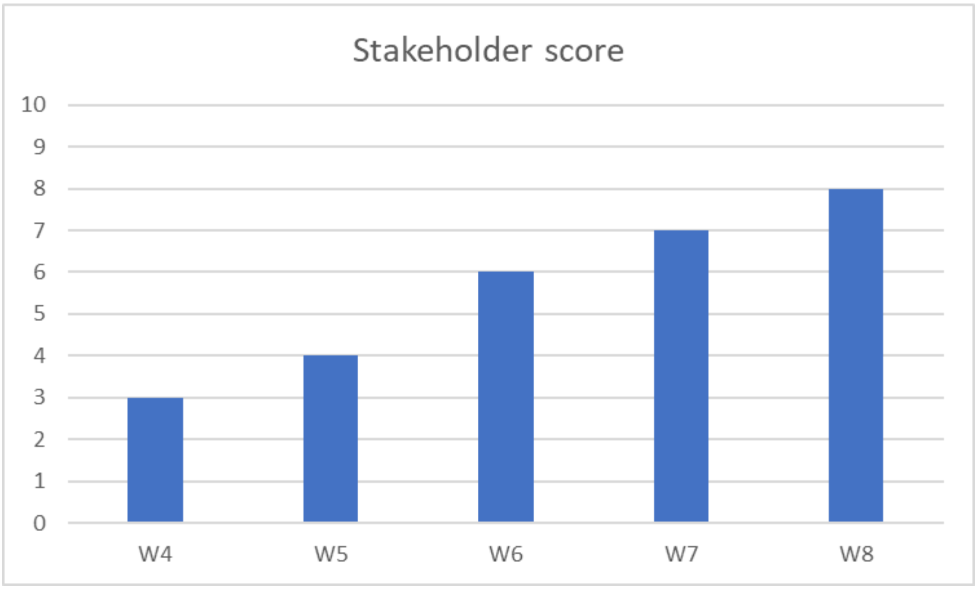


Figure 2: Stakeholder score between week 4 and week 8 of the project.

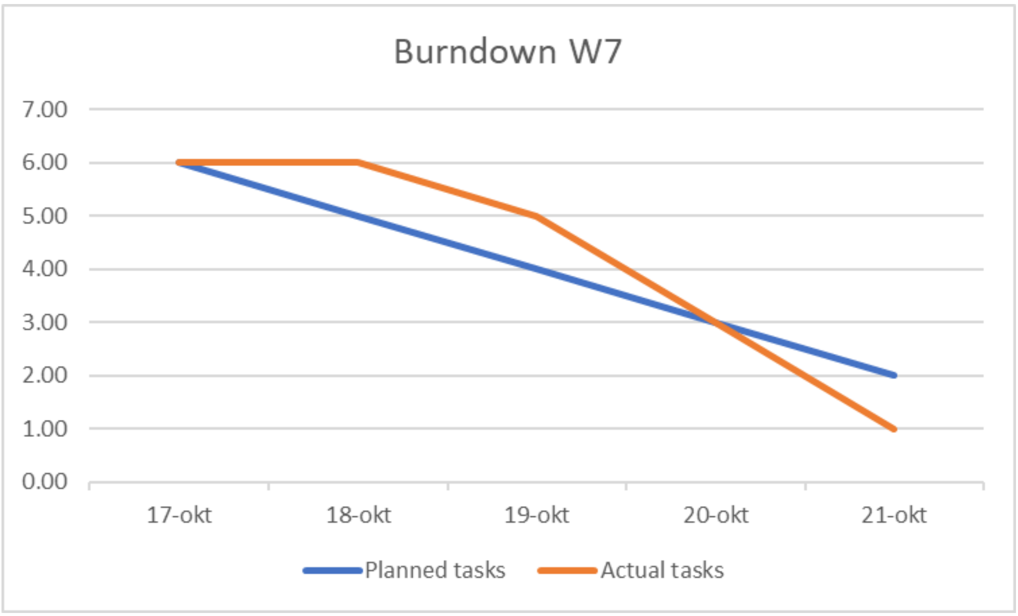
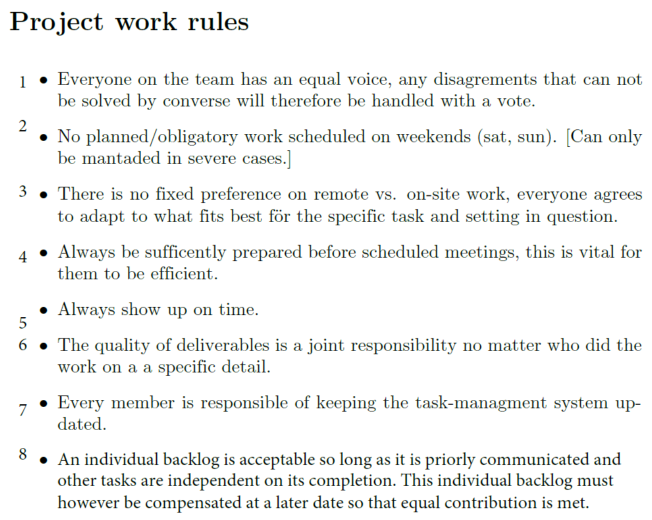


Figure 3: The burndown chart for week 7 of the project.

2. Social Contract and Effort

2.1. Social contract

A: After coming up with our initial project idea, we set out to form a social contract (Yavin, 2022b) applicable to all members of the team, detailing a set of rules that we all agree on in terms of the project, with the purpose of facilitating an improved work structure. It also increases the level of accountability that can be asked from a given team member. The contract detailed routines regarding internal team hierarchy and decision-making capacity, the work-life balance and remote work aspect, courtesy rules such as meeting etiquette, etc., and routines involving task handling and ownership. In terms of iterations of the contract, we have only completed one amendment.



Project work rules

1 • Everyone on the team has an equal voice, any disagreements that can not be solved by converse will therefore be handled with a vote.

2 • No planned/obligatory work scheduled on weekends (sat, sun). [Can only be mantaded in severe cases.]

3 • There is no fixed preference on remote vs. on-site work, everyone agrees to adapt to what fits best för the specific task and setting in question.

4 • Always be sufficently prepared before scheduled meetings, this is vital for them to be efficient.

5 • Always show up on time.

6 • The quality of deliverables is a joint responsibility no matter who did the work on a a specific detail.

7 • Every member is responsible of keeping the task-managment system updated.

8 • An individual backlog is acceptable so long as it is priorly communicated and other tasks are independent on its completion. This individual backlog must however be compensated at a later date so that equal contribution is met.

Figure 4. Project work rules as stated in Social contract (Yavin, 2022b)

Pretty much all points of the contract have been followed by the team throughout, where the more formality-characterized points (see 1-5 in fig. 1.) have been easier to always live up to. In contrast to the more active work-related rules (see 6-8 in fig. 1.). The benefit of firmly agreeing on things that might seem trivial and expected at first, like voting rights, showing up on time to meetings, etc., is that it helps us to manage expectations within the team and forms a bedrock to benchmark our intra-team behavior against. The more tangible work-specific rules have been harder for us to always live up to, and it was also here we had to iterate and make an amendment. Specifically, rule nr.8 (see fig 1.) was added after realizing that individual team members might end up in a situation where a task deadline has failed to be met. We decided on implementing a system where we allow this potential individual backlog to occur, so long as the member communicates this risk clearly and makes sure that it will be taken care of in the following sprint. The reasoning behind this was that we wanted to ensure equal contribution across the team, so that one person can't fail to meet his obligations and then let someone else pick up the slack the following week.

B: All in all, we argue that the contract was well-suitable for us as a team, and we see the value in forming a concrete universal understanding of expectations, etc. However, we have come to the realization that it is not really what is stated in the contract that we should reflect most about, but more reasonably, the aspects that we did not take into account. In retrospect, we have concluded that we probably should have iterated more and amended the contract further during the project. Due the fact that 80% of the group have known one another for a long time, which has, at worst, made the work less focused and at best very seamless since we all are used to talking with each other, however, this close relationship has made it difficult to enforce any form of punishment, etc. when a given rule has been broken. In the future, we feel like we would in this type of team setting need better accountability structures.

A → B: One example of a rule or routine that can help facilitate this improvement is one that emphasizes the importance of having an enforcement mechanism in place when a given rule etc. is not followed. It should be clear what the non-negotiable consequences are when a given principle is not lived up to. We think that it would be very beneficial for us to come up with some form of structure that "enforces" the rules and makes sure that we can't just laugh a deviation off. One example could be paying a symbolic "team-tax" when a rule is broken and the team can later use that money to do something together outside of work etc. Nevertheless, no matter what tool one uses, we see the importance of assuring the presence of accountability and enforcement mechanisms.

2.2. Effort and time allocation

A: The time spent and workload of the project has varied a little bit over the different project sprints due to various reasons. The initial stages of the project were characterized by a lot of planning and figuring out what we wanted to do etc. One major mistake we made was that we implemented a different sprint structure, other than weekly sprints, which led us to basically rework the entire sprint structure after the two first weeks. This created some initial problems but after feedback from our supervisor we felt like we turned it around to get on the right track. The middle stages of the project were the most productive and also the stage where most tasks could be better divided among team members, facilitating higher efficiency when people can really work individually in the future while someone else works on another area.

We have during the project kept track of the time spent on each given task in terms of directly product-related work (i.e. coding, data gathering, design, etc.). This was later summed up to 111 hours. The distribution of time spent across the project was relatively stable however, some spikes were noticed in the beginning and in the end of the project, firstly when we were figuring out what to do and secondly when it was time to tie up loose ends and make sure we met the MVP goal. However, we didn't keep sufficient enough track of the time spent on indirect work, i.e, tasks related to the project but not directly to the product such as meetings, reflections, etc. We estimate this amount to be close to the same amount that was spent on the directly related tasks.

B: If we were to do this type of work again in the future, perhaps in a business setting, we understand that it is paramount to keep track of all hours spent on a project, no matter if they are billable to the client or not, but in particular when they are. Furthermore, we want to have a more even distribution of required time across the sprints of the projects, i.e, reducing the prevalence of spikes. This will, in turn, make the project work a more pleasant experience overall since the team member stress level will be reduced.

A → B: We, therefore, wish to form a solid time-tracking system for all aspects of the project from the start so that we know that we have spent a lot more time on something than what can actually be measured in retrospect. We also think that a better time-tracking system will facilitate a more even distribution of time spent in each sprint.

3. Design decisions and product structure

3.1. Design decisions

A: In the beginning stages of our project, the design decisions were very surface level, and it mostly relied on our presentation mock-up, which included a table of the data and a graph to showcase it. We decided on using Java-Swing as an API for providing a graphical user interface, this worked well since most of us had worked with Swing before. We wanted our program to be low in coupling and high in cohesion and therefore decided to structure our project after Model-View-Controller (MVC) architecture.
We also decided to use some open-source APIs for pulling data about election voters and municipalities. In the very beginning, we realized one of our open-source API:s were empty, and we had to look for a new one to get the data we needed. Luckily we found a very good open source data source that included multiple other datasets which we later could expand on. After acquiring a stakeholder, it was also discussed how we needed more datasets and that the graph was very important for the final product. That same week the project group also decided that it would be good if the user could search through the data. This was in later weeks achieved, we expanded our dataset, introduced a graph, and made a search function available.

B: One change that would make a big difference is switching out Java-Swing for another graphical user interface. One of the reasons why switching would result in a better design, is that Java-Swing does not really support full MVC structure, and it is relatively old, which makes the program look outdated.

A → B: As was stated before, replacing Java-Swing would improve the design of the program. This could either be achieved by using JavaFX, which fully supports the MVC structure, or we could have created a website using something like .NET, which also supports the MVC structure. Both these options would greatly benefit not only the structure, but it would also help the program look better. Since our project idea relied on data and we struggled at the beginning with a faulty API, double-checking the contents would have saved us time.

3.2. Technical documents and how we use and update them

A: Our documentation was mostly javadoc-comments made in the different classes, which served its purpose of making others understand how and what the classes are used for. But we also agreed it was lacking, when there were some classes without javadoc-comments. In later stages, we ran upon the issue of not everyone in the group understanding how classes interact with each other, and we then decided that it would be a good time to introduce a UML diagram. The UML diagram was created and upheld for a couple of weeks. It also served its purpose and helped everyone understand how the classes interacted with each other.

B: Creating a UML diagram earlier on before we even started writing code. This would have helped our program structure. It would have also given us a quicker start because we would have already planned the classes and functions needed to achieve the functionalities that we aimed for. Our Javadoc-comments were of relatively high quality, for the most part, however, we did slack in commenting on our classes after a while. Having a way to ensure that the Javadoc-comments always exist for each class would ensure good documentation. Since the program was relatively small, adding more documentation than a UML diagram and Javadoc-comments would not add further utility.

A→B: As stated before, creating the UML diagram earlier would have given the UML more use cases. Instead of previously only explaining the existing code, creating it early on would show us what to create, and how classes all were connected. Then to have all the classes include Javadoc-comments, we could add the Javadoc-comments as an acceptance criterion before the pull-request merge. This would help us keep the classes' documentation up to date.

3.3. Code quality and enforcing coding standards

A: At the beginning of the project, we had no specific way of enforcing coding standards and code quality. We later introduced branching and the usage of pull requests, which introduced a checkpoint for the code to be reviewed before being merged with the main branch. We feel like this worked fine, however, there were some instances when we did not make a pull request and merged straight to the main branch. But this occurred mostly during times when we were coding in groups which meant we checked throughout the process of coding.

B: Starting out by having an assigned "pull-request reviewer", someone who was in charge of the code and overlooking the quality. Choosing a standard for naming variables and functions. Increasing the overall quality of the code, better complexity and better solutions to certain problems.

A→B: Having that extra role of "pull-request reviewer" would assure that the full code all be overseen through the same lens, and overall produce a better coding standard. Picking a name convention, such as lower camel case, and assuring that all classes use the same one, makes it easier and better-looking code in general. Making tests to assure that classes are up to par with the goal in mind.

4. Application of Scrum

4.1. Our Roles

A: We took on a classical role division of having a Scrum-master, and one role for communicating with our stakeholder. The scrummaster was, in our case, responsible for adding tasks to the scrumboard and setting the preliminary agenda, as well as guiding the conversation in the continuous retrospectives, reflections, and reviews. Our communicator was primarily responsible for that stakeholder sessions were set up and that the agenda for those sessions was aligned with the team's progression and aspirations for the project.

B: Considering the alternative approach of having more roles, there could be potential benefits such as better division of labor, and better overall understanding in respective domains. With the potential negative effects such as siloed thinking and increased need for communication on administrative tasks. So, in the best case, better assignment of responsibility amounts to more of the administrative work being evenly distributed amongst team members and would likely result in a better overall understanding of these individual parts. In the case where there is one person responsible for the scrum board, it easily becomes the case that other team members are less inclined to utilize it optimally, especially when no one has worked with one before. Something we, to some degree, initially experienced. However, in a more experienced environment, the team members are more familiar with the use of the scrum board. As such, team members likely contribute to it with more ease.

A → B: Given the alternative to our approach is contingent on less experience in the team, our role set-up is likely better for future projects. The downsides of having only one person responsible for the agenda and scrumboard are likely reduced with experience in the general team.

4.2. Agile Practices

A: Semi-daily scrum meetings: We have, with increasing frequency, utilized daily scrum meetings to align the team on progress, as well as issues. And to draw value from each other's experiences in prevailing issues. We opted to increase their frequency during the progress of the project as we experienced issues with double work, which we attributed to the unanticipated speed of certain members and a lack of alignment in the scope of each other's work. After the increased frequency, these issues did not occur again.
Scrum-board: We applied a generally classic scrumboard containing: User Stories + Continuous tasks, Product-Backlog, Sprint-Backlog, In progress (tasks), to-be-reviewed(tasks), and Completed(tasks). Generally classic due to our addition of continuous tasks. Continuous tasks were incremental tasks native to the nature of the project, being a school project with overhead tasks such as the submission of reflections. This was somewhat convoluting to the scrumboard. In other aspects, we felt the divide felt natural. A brief explanation of our use of each column follows:
User Stories + Continuous tasks: as mentioned, continuous tasks where those associated with this project being a school assignment. User-stories is where each sprint had their

respective userstories written out in the same format which they where communicated with our stakeholder.

Product - Backlog; This was a catchall for tasks not completed during a given sprint but it also turned into containing tasks for the project not directly associated with any one specific sprint, but still important to keep track of.

Sprint-Backlog: This is where we created all new tasks, ergo the first step in the process of completion. Tasks could be formed here, developed and if not completed transferred to for example product backlog. But generally, the tasks progressed into progress, review then completed.

In progress tasks generally had an assigned teammebers responsible for its completion. The same member would then move it to To-be reviewed.

To be reviewed, was our primary column of interest in before completing tasks. This is where are DoD was enacted. We, initially only reviewed tasks two and two and accepted as such. However, we realized the benefit of collectively assessing the contributions. This resulted in a better understanding of the workings of our contributions.

B: Daily scrum meetings to ensure optimal work load for each team-member. The scrumboard should at all times be an accurate representations of past present and future state of the project, with enough information to easily work independently. The categories on the scrumboard should be clear, mutually exclusive and collectibley exhaustive.

A → B: In order to achieve a daily frequency, the only thing needed is an aligned understanding of such a setup going into the project. We also would need more time allocated to the project. As we all had other contingencies we could not realistically find time each day for these sessions. So another precondition would be less fragmented schedules. As previously mentioned, experience in working with the tools becomes a determining factor of its success. As such, the scrumboard will have increased utility when it occurs more as second nature and less as an obstacle. We managed to overcome it as an obstacle during our project, however, learnings from the end state of our use would be need to be utilized in early definitions of its usage. As with much in agile practices, planning is key. We did not consider our scrumboard before we started using it. Therefore, some effort into its categorization and a teamwide understanding of each category's use would facilitate a better utilization.

4.3. Sprint Reviews

A: The sprint review was as aligned to our scope and customer value as we could manage. The conversations where as previously mentioned predefined by the scrummaster with alterations welcomed in advance to the meeting having started. Otherwise, if new topics arose, they where quickly assessed as top priority or low priority. In the case that the proposed agenda point was contingent to other agenda points, it was deemed to be a top priority and thus handled first. If not, the agenda point was added to the end of the meeting. A typical review would consist of the following questions along the lines; what considerations do you have? what do you like about our progress? What would you change if you did it again? What is your worst aspect of the product right now? Our retrospectives were focused on our teamwork with the following questions; What went well? What didn't go well? What are the implications of these feelings for upcoming teamwork? Learnings from the past sprint? Besides these sessions, which in many cases was associated with the production of the deliverable "Weekly Team reflection" we had meetings with our stakeholder, sometimes, all of us, but primarily our contact person. In these sessions, we initially did not have much structure. We did not get technical on the product, but we could have had more structure to make sure all bases were covered. We did have one KPI linked to our stakeholder satisfaction which was mentioned previously, this helped us with some reference for how we were progressing.

B: It's important to give more time to the sprint reviews and retrospectives. Make sure they are clearly separated, they are similar in questions, however, they ask about two very different and equally important aspects; the product, and our teamwork. Therefore it is important that both are allotted equal time, and that the distinction between the focus on the product and the teamwork is clear for all team members.

Having a planned structure for meeting with the stakeholder facilitates greater utility in the feedback from the external part. You are also better able to track the progress if each week is comparable in further aspects than just an unexplained KPI tracker.

A → B: As with many aspects, we consider time and experience to be determining factors of success in agile. The more time you give to planning and especially to structure, the more you win in the long run of the project. While these plans should always be open for iteration, they allow for time saved and, in the case of utility, additional benefits such as future value. For example, with stakeholder meetings, better planning of the structure would result in more useful conversations and the ability to compare progress. This theme is also apparent in the case of the sprint review and retrospectives, but mostly in this case. The determining factor is experience. So far in school, we have not been exposed to these types of conversations. To openly reflect on ourselves and our team members, is, in some cases, uncomfortable, and having more experience in doing so will allow easier adoption of such practices in future cases.

4.4. Best Practices

A: We utilized git for much of our toolset, both in terms of scrumboard and also for our version control. Some additional tools were the UML creator; DrawIO as well as Intellij for code production. We were all familiar with IntelliJ since previous work, and no real expertise development was tangible from any group member. However, in terms of version control, we all developed immensely. Previously most team members had not utilized version control since there was a perceived complexity innate to the process and since much of previous collaboration was in pair programming, which in large eliminated the benefit of version control in their understanding. In terms of scrumboard, we, as mentioned, noticed a development in our proficiency. We made advancements in both correct usage and overall usage during the project. However, we did not manage to utilize it to the extent that we could draw benefits from the additional functionalities, such as tracking the progress of tasks and other aspects we required for calculating weekly KPI progression. This resulted in much time spent on the administrative tasks of manually calculating these.

B: Agile projects, maybe all projects, are completely dependent on great use of version control. Without version control, and especially branching, the process would have become exponentially complex with a growing codebase. Something we discovered early. The scrumboard functionality is completely utilized, and as such, the additional benefits of having one on github are realized, such as statistics on task completion, etc. Furthermore, the best use of tools arises from a well-defined need. Along with a good scope of tools aligned to solving this need. In the event that the teams need to extend their toolsets, a process for defining, and selecting the appropriate tool is a way to make sure successful and purposeful integration is achieved.

A → B: We need to define a process for selecting tools, while we managed to get lucky in our first selection in this case, with, for example, the scrumboard on git, we might have had an easier time learning to use it if we, for example, had a selection criterion in the simplicity of the tool, we might have needed more guidance in how to optimally use it.

4.5. Relation to literature and guest lectures

A: The main aspect of literature and of what Matthias Becker noted is the importance of planning. This, and our previous inexperience with certain methods, have really been the basis for most parts of what we would have improved if we were to redo this course. We paid, by not conducting a qualitative planning phase, in problems down the road in the sprint structure, KPIs, the usage of the scrumboard, and the sprint reviews during the end-of-the-week meetings. In retrospect, all of these weekly slices involved when doing a team project would have been much more effective if we would have had a greater quality in our planning phase.

B: With teachings from both the course's literature and the enlightening guest lecture by Matthias Becker, the major point of reflection that hit our group the most was the lack of the planning stage, as earlier mentioned. For future projects, this is a part that we will spend much time improving, which will hopefully yield much better weekly progress and team- and stakeholder satisfactions.

A → B: As earlier mentioned, to get to a qualitative planning phase, we will spend more time planning the actual structure of the project and hopefully choosing comprehensive KPIs to measure all important aspects of the project's progress.

Overall Conclusion

The course has served as an important lesson to all team members in learning more effective ways of working together towards a common goal. Both in terms of communication with weekly meetings, but also in working independently without any unnecessary mishaps along the way. On an individual basis, the course has also evolved each team member in different ways depending on each member's previous inexperience in certain areas. The main overwhelming learning that each of us will bring with us is specifically the importance of the planning phase. Here we wished that we would have spent more time to cover all areas that we later would end up working with. For example, spending more time reflecting on what KPIs to use to, in turn, receive the most effective feedback, or how we would like to structure the weekly team meetings with a predefined agenda.

Despite difficulties along the way, our stakeholder ultimately gave us a stakeholder satisfaction score of 8 out of 10. This shows how we reflected on our mistakes, and improved over the course of the project, as we initially started on a stakeholder score of 3 out of 10. The initial score was based on the stakeholder's perception of how much value we had created thus far in the project versus how he would have preferred in an ideal world. Therefore, the increase in stakeholder score serves as a great signal of how we improvised and improved our work over the weeks of the project. This was partly driven by the weekly feedback from our supervisor, stakeholder, and reflections from what we ourselves had understood were inferior ways to work. To conclude, the course really lived up to improving each of us and our abilities to better structure future projects to yield more valuable outcomes.

References

Yavin. (2022a). *Project Plan Yavin DAT257*. Available from:
https://github.com/users/lidvall/projects/1

Yavin. (2022b) *DAT257 Social Contract: Group Yavin.* Available from:
https://github.com/lidvall/yavinDAT257/blob/main/methodology/social_contract_signed_all_amended.pdf