tRAINing - a Jakku project

Ludvig Andersson
Oliver Brottare
Christina Meisoll
Liam Mattsson
Erik Gustavsson

Introduction tRAINing project

The purpose with this project was to give us experience with software development from an agile approach. In this report we discuss our project from the perspective as developers to describe what we wanted to do, what we wanted to learn, what we did and what we could improve in similar future projects.

Project Idea

The idea behind tRAINing was to create an android application that would help the user with planning study hours, training hours and getting travel recommendations based on weather.

Customer Value and Scope

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

The scope chosen allowed us to have a reasonable amount of functions with a reasonable level of depth to them. Due to the way the app was built, everything was made to create value for a user who uses all the parts of the app, with priority mostly based on getting related parts done first.

The different types of users were not always clear, most user stories were listed as “as a user”. Clearer user stories in that sense could potentially help see how many users a function adds value to, and prioritize based on that.

This change could be made by talking more about the user stories early, discussing how many people a user story adds value to and prioritizing based on that.

The success criteria for the team in terms of what you want to achieve within the project (this can include the application, but also your learning outcomes, your teamwork, or your effort)

We wanted to learn how to make an app and hopefully manage to add all the functions we set out to have from the start. We also wanted to improve our use of scrum, primarily focused on making decent sized user stories, and also try to ask for help when we get stuck.

Towards the end we got a bit better at asking for help when needed, however we could have been better at it, especially in the earlier stages of the project. We could also improve at checking up how everyone else is doing, although that also improved towards the end.

In order to more easily see who might need help with something we could improve our use of Trello. More specifically, splitting user stories into more tasks and properly marking which tasks are done makes it easier for the rest of the team to see who might be struggling.

Your user stories in terms of using a standard pattern, acceptance criteria, task breakdown and effort estimation and how this influenced the way you worked and created value

When creating our user stories in Trello, we set up user criterias for each and everyone of them. Everyone in the project group discussed what each acceptance criteria would be, and the person working on that specific user story could then, after it was done, show it to the team how it works and that way prove that it fulfills the acceptance criteria. After making sure it was working, we then showed it to our stakeholder to make sure that he also accepted the way it was working, since he theoretically is the one that's gonna use the product later on. On every user story, we tried to estimate the effort by using the built in story points in trello. We didn't use it on all of them, since we in the beginning didn’t know about the point system available in Trello, so in the earlier phase we only discussed estimations on user stories instead of documenting them. The way we distributed the estimation points was that we discussed in the sprint planning what estimation value each user story could possibly have.

Discussing effort estimations as group was working well in this group constellation, since we have worked together before, have trust in each other and are confident in discussing different opinions openly. For the future though, when we work in other groups it will probably be preferable to do the estimations as poker again in order for everyone to feel comfortable.

If there were anything we could have done differently to make it better, we could have searched for similar apps to Trello to see if they would have an option to individually estimate the user stories. Another option would be that the examiner would have a lecture where they quickly present trello and other such apps so that we don't need that much research about the apps, but also having more options to choose from to use one that fits our group the best.

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

Acceptance tests were primarily done by the one developing that part. This was supplemented with some discussion within the team and then finally showing it to the stakeholder. This worked pretty well, since the acceptance criteria were set up by the team.

Perhaps we should have reviewed each other’s code more often and discussed it. This would have made it easier to understand the code when adding new parts, and also made it easier to find bugs in the code faster.

This would be accomplished simply by having more code reviews. Having it one-on-one would be an option, but since the team is pretty small, having team-wide reviews where everyone takes turns showing what they’ve done during the sprint would be an option. Also if we in a future project would work with something more function critical we would do more excessive testing.

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

The KPIs were mostly focused on keeping track of how much work we've done and how much is left. This helped us even out the work over the weeks and make sure we were on track. The scrum master planned accordingly as well and tried to make sure that we had at least 3 user stories to work on each sprint.

The KPIs were not set up during the first day, the sooner they are set up the better. They were also not shown to the stakeholder during the project, which could make it clearer for the stakeholder how the project is coming along, rather than just showing what parts are done.

In order to achieve this, we would simply set up the KPIs immediately after we're done making user stories and show the KPI charts to the stakeholder during the review.

Social Contract and Effort

Your social contract, i.e., the rules that define how you work together as a team, how it influenced your work, and how it evolved during the project (this means, of course, you should create one in the first week and continuously update it when the need arrives)

When creating our social contract, we took in consideration that all five team members have worked with each other in previous projects. Still, we found the social contract to be useful to communicate expectations from each other and especially communicate working time boundaries. Things that in previous projects sometimes have been unclear. For instance we stated "Everyone else is off on weekends". This was a good point to have in the contract since it clearly stated what the mindset should be, to not expect someone else to work on weekends, which in some cases might have been difficult to communicate to each other earlier. We had more similar points that were good for laying the foundation of how our project was to be executed.

One point we initially didn't follow through in the earlier stage of the project was point 6 "When you get stuck, you consult your teammates after ½ day". We realized that some team members were done quite fast in a sprint and others didn't get anywhere, which we realized on the last day of the sprint. The agile approach here helped us to adapt and we decided on a team meeting that we were to improve on this which we did.

Another point in our contract we would like to improve is "Everyone documents their code in a way so their teammates understand". In reality we didn't document much on the go, instead we explained to each other at each meeting. On our next project we want to make sure everyone writes a couple of lines on each method/class, before each commit. This is to help team members understand the code without having to remember or guess how it works.

The time you have spent on the course and how it relates to what you delivered (so keep track of your hours so you can describe the current situation)

Due to deadlines in other courses we sometimes had less time to spend on the project and we tried to plan accordingly when picking user stories. Overall we tended to spend roughly the same amount of time each week. We managed to mostly deliver what we set out to each week, however a large chunk of the time spent on the course was spent during meetings and reflections.

It felt like less time should have been spent on meetings overall, which could mean more time spent on for example code reviews. Part of the issue might have been that we sometimes did some group programming during meetings with no clear boundaries.

More structured meetings could help in this. By keeping the whole team focused on one thing at a time we could potentially use the time more efficiently, and hopefully either decrease the time spent on meetings or increase the amount of work done during the meetings.

Design decisions and product structure

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

Since our plan was to create an Android app that provides information to ease their life, provided by the weather, it was natural for us to use an API that provides weather information. There were a lot of global weather API:s that we could use for our app, but since our main audience for this app were people in Gothenburg, we wanted to use a more local API. Our choice then came down to using SMHI:s own weather API. Since SMHI is based in Sweden, it gives a better value for our customers since they are based in the same country. To get some structure in our code, we made the decision quite early to try to code according to the MVC design architecture pattern. This was before we really had any knowledge about Android Studio. During the development of the app we realized that this is gonna be much harder than we thought since Android Studio provides a lot of different things than regular Java, which makes it harder to stick to a certain pattern. However, we still tried our best and made classes that were specifically made to handle only data, for example.

One potential way to improve would be to choose an architecture pattern early and stick to it. It’s possible that this wouldn’t change much, but potentially it could make it clearer what needs to be done and therefore speed up the work process.

In order to choose an architecture pattern we would have to do some research on what patterns are used when developing android apps, rather than attempt MVC and realize it doesn’t really work. One option would be to look more into model-view-viewmodel.

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

We mostly didn’t have that much technical documentation. When working on something that required understanding of someone else’s code, this was mostly done by reading the code and asking the person if something was unclear. We did make a class diagram eventually, but didn’t add to it after that. The reason why we didn’t have much technical documentation is because we haven’t worked with apps before, so we felt unsure what we could even plan to do.

Our use of technical documentation could easily improve by putting more focus on making it and using it. Although it worked pretty well without it, a clear class diagram earlier during the project could potentially help during the project.

We could have started the project by making a class diagram, even though it wouldn’t be perfect. This would most likely require some heavy changes during the project and take up some time, however it could potentially be worth it since it would make us try to understand how the platform works with a bird's-eye perspective.

How you use and update your documentation throughout the sprints

We haven't kept any document or so where we document our work other than code comments, team reflection, sprint retrospective and sprint supervision. Our documentation has been split up into many different parts where we used discord for planning meetings and sending potential errors in the chat to help each other. Trello was used as our scrum board where we added user stories that we distributed among us. We also had a google drive folder where we added our team reflections and supervisions. One thing we were missing was documentation of our meetings. Our first visible documentation of the project was when we discussed our first prototype of the app. This prototype is available on the GitHub page. Other types of documentation we have is our own weekly individual reflections, where we describe in detail each week how our work is going.

Something that would have been better would have been to document our meetings and our conclusions because most of it just became verbal but never written down anywhere. There is still some documentation in Discord as well that's in one big channel so trying to find old messages means you have to scroll through it all. One thing that we also didn’t really document was the meeting between the PO and stakeholder. Some of it was written down in Discord but not all.

For the next project we could have a detailed project protocol where we write down our conclusions of the meetings to easier see what we’ve done each week. Documenting the PO and stakeholder meetings would also have been an improvement. Discord can also be divided up into multiple channels where we can have more technical issues in one channel and other discussions in the main channel.

How you ensure code quality and enforce coding standards

We didn’t ensure code quality very rigorously. It was mostly up to the one who made the function to test it and make sure that it worked, code review was done sometimes but not very often. We also didn’t set up any clear coding standards, but that didn’t lead to many issues, probably since we mostly worked on our classes alone.

In order to better ensure code quality and coding standards, we would have to interact more with the parts of the project that we didn’t work on ourselves.

This could be done simply by having more code reviews as stated previously. We would have to set aside some time during one of the weekly meetings and look through what everyone has done so far.

Another possibility would be to agree on one procedure when branching off and merging, so the branch of code always is on the most recent valid version of code and before merging all conflicts could be handled locally. This would reduce the occurrence of bigger merge conflicts and thereby reduce the risk of old and new parts getting mixed up.

Application of Scrum

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

When choosing our roles we choose to have Christina as our scrum master and Ludvig as PO. Ludvigs father became the stakeholder so Ludvig and him were able to communicate easily about how the project was going and talk about prioritization of features. Erik, Liam and Oliver were developers. The impact of our choice was that Christina communicated with the team and helped organize the work. Ludvig communicated with the stakeholder and then told us about the stakeholders thoughts about what we were doing, which led to some reprioritization. Thedevelopers didn’t do much unique stuff other than what everyone else was doing, which was coding and participating in meetings, which was their job.

We believe that our choice of roles was very good, if not optimal. One other thing we could have tried was to pass around the roles, having different scrum masters for each week for example. This might not have been better but it would have been different. We do not think this would have a big impact on the project however.

Doing this change wouldn’t have been hard since we would just shift responsibilities. The more important and hard bit would be to watch out so that the quality of the project would end up worse than before.

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

We used the scrum method with weekly sprints as our agile approach for this project. We began each sprint with a sprint planning meeting each monday physically on campus. In this meeting we planned our user stories. Earlier in development we discussed what user stories to have and also what tasks there could be on each user story. Later in development when user stories were set we used the sprint planning phase to choose which user stories to work on. We did this by estimating difficulty for each user story and guessing how many user stories we could manage during the sprint. During the sprint we had “Every-other daily scrum meetings”. This is since we felt we couldn’t always create or get value each day since we might be occupied with other courses for some days and therefore make daily scrum meetings redundant. So in practice we had our sprint planning on monday, “daily” scrum meeting wednesday and sprint review on friday. So monday and fridays also worked as “daily” scrum meetings. During the Friday meetings we did some team reflections and later on retrospectives which made us think through the project more on what we’ve done and how we would do the rest. It made us discuss the project more and made us reconsider some user stories. Retrospectives helped us to identify hinders.

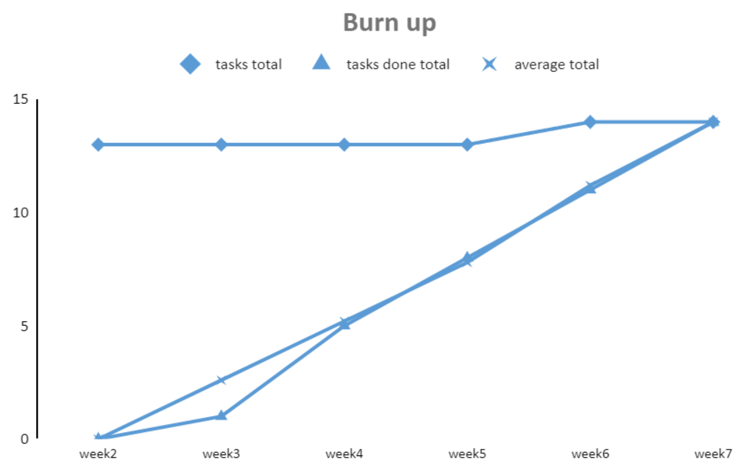

Figure 1. Burn up chart from our KPIs

Since working with this method we have managed to maintain an even and sustainable pace throughout the course, as can be seen on the Burn up chart above [1]. In a similar project we worked on last year we didn’t work with scrum or any other agile method, making some of us work at an unhealthy pace during the last weeks of that project. This approach also helped us maintain a minimalistic work scope, since by using scrum, we automatically had to follow the user story pattern which doesn’t really allow unnecessary deflections from the minimal viable product, if used correctly.

The implementation of scrum didn’t work like clockwork from the beginning, but during development we learned how to use it better and better. For instance we first created a backlog with user stories and user stories tasks that looked the same but with additional offset for the tasks. Ex: User story 1.0 and tasks 1.0.1, 1.0.2. We also didn’t write the acceptance criteria. After the first sprint we realized it was hard to differentiate tasks from user stories. It was also hard to determine when a user story was done. We also created too wide user stories from the beginning. Some of the tasks could have been made into user stories instead. On the next project we will use this experience to have a more efficient usage of scrum.

Another thing we could do better is to separate the “daily scrum” part of the meetings with the sprint review/retrospective/planning when they occur on the same day. We were sometimes not so efficient here since some team members started to work on the team reflection while some team members were programming at the same time. This made some have to wait and do nothing while others were programming, which sometimes could take a lot of time.

On our next project we will have a clear “coding part of meeting” and a clear “Coding forbidden” policy during team reflection etc. This could help us to actually put down the code for a while (which sometimes can be hard when you’re really irritated by a bug etc.) and also help us have a clear agenda of what the current topic is. Other than this, we were satisfied with how we used the scrum method.

The sprint review and how it relates to your scope and customer value (Did you have a PO, if yes, who?, if no, how did you carry out the review? Did the review result in a re-prioritisation of user stories? How did the reviews relate to your DoD? Did the feedback change your way of working?)

Our PO Ludvig showed the stakeholder, his father, what we had at the end of the sprints. Sometimes the feedback resulted in some added features or reprioritization, mainly the addition of gps to the app, but mostly the stakeholder thought we were keeping a good pace. At one point we had to change the DoD, since we realized that checking for multiple phones/multiple android versions would not be sustainable, which the stakeholder agreed with.

The sprint reviews could potentially be improved by having everyone talk to the stakeholder, not just the PO. It would probably give the stakeholder a better view of how the project is doing and what our plan moving forwards is.

One option could be to properly schedule a meeting with the stakeholder, which would allow the whole team to participate, instead of talking via the PO.

Best practices for learning and using new tools and technologies (IDEs, version control, scrum boards etc.; do not only describe which tools you used but focus on how you developed the expertise to use them)

The biggest new thing we had to learn about was Android Studio and creating apps on an emulated phone. Android Studio is very similar to IntelliJ which we had used before so the interface was similar. Learning how to emulate a phone and develop apps for it was mostly done through guides and asking each other for help. When others had done something you could test and read their code to get a deeper understanding of how it worked.

We felt like this was a good way to learn and never saw any reason to change it. It is always possible to try and tackle any problems together to learn together but doing so could also have slowed down the project a lot.

With that in mind, as mentioned earlier, trying to create a UML diagram of a program on a new and unknown platform could potentially help us to understand a platform from a broader perspective. There is otherwise a risk that the group “trial and errors” a program, thus implementing it in a wrong way leading to headaches later on. So to spend a couple of days learning a platform first could potentially be helpful, but often learning by doing works just as good.

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

Throughout the project we felt that there were a couple of things the speaker said during the guest lecture that we could relate to. First point was that at the beginning of the project it was harder to define and make estimations on user stories but throughout the project it became easier. As we got more experience with the platform, with scrum and the collaboration between team members this got easier after hand. Another point we recognized was that it can be a bit hard to show something valuable to the PO when all the work you have done in a sprint is backend. Now in our case, the backend programming most often had clear usage making it easy to explain why it provided value to the PO.

The guest lecturer also mentioned that they had a strong review culture, making reviews high priority. This is something that we could try to incorporate a bit better, as we have stated previously in this text. A stronger review culture could be achieved by having time specifically scheduled for code review during the week, rather than doing it sporadically as we have.