Post Mortem Report

Skrubbgruppen

Intro

H-sektionen is a Android application that gives all the students easy access to information regarding H-sektionen at Chalmers Lindholmen. This App was developed by 6 students as a school project. We used the following processes and practices in our project:

- Scrum
- Git
- SeeNowDo
- StandupMeeting
- Sprint-planning
- Backlog-planning
- Gitflow
- Working close together
- MVC-pattern

Time estimation

Every week started with a sprint-planning meeting (tuesday) with all participants in the group, the meetings lasted for about one hour.
We had programming sessions in group about 3 times every week where everyone would come to program and discuss the project.
We started each programming session with a scrum-meeting which lasted no longer than 10 minutes, where everyone said what they had done the day before and what they planned to work on for the rest of the day.
We spent about 126 hours per week working on the project spread over the whole group.

An estimate about the different activities we've spent time on for the project per week and person:

- programming - 17 hours
  - researching - 50%
  - testing - 20%
  - actual programming - 30%
- meetings - 1 hour
- planning - 1 hour
- documenting - 2 hours

Processes and practices

Scrum

We enjoyed working with Scrum. The Scrum method meant that it became easy to follow the work progress of the whole group. By breaking down the project into small user stories it became easy to create tasks that was of the right size to be completed in a few hours. That helped in the weekly planning by allowing us to better understand what to prioritize and to get things done. It also made it easy to make the best use of the workforce.

In the project we did try pair programming for the first time and it was both good and bad. The benefit from it was that it made it possible to help each other in trickier situations. But sometimes, especially in the beginning when we did not know anything about Android programming, we just wanted to spend time online researching and learning about how to code specifically for Android, so it felt more inefficient and unnecessary to do these things in pairs.

We liked working with different roles. By having a Product Owner assigned to the project we were able to move along faster by not getting caught in long discussions. It was good to have a person with a veto. Our Scrum-masters role was to lead our daily scrum meetings and sprint planning. We all benefited from the daily scrum because it made everyone more involved in the project. The sprint planning was in an open forum but with the project owner having the final word.

The burndown-chart was a good way to follow up the weekly sprint. By committing hours to the burndown chart the members of the group had a good overview of how the work process was progressing. Sadly we did not always update the chart on a daily basis, which resulted in the burndown chart not being 100% accurate. We also think that we could have tried out “planning-poker” more and also followed the definition of ”done” in a better way, by releasing only after proper testing.

We think that Scrum is well suited for groups of developers making software close to the customer, with small releases close together. But if working alone or in very small groups we found the benefits of Scrum decreasing.

SeeNowDo

To help plan and keep track of the different tasks for the weeks sprint, we used an online program called “See now do” (https://www.seenowdo.com), which allowed us to create tasks and give them values depending on the estimated time for completion. The tasks could then be changed to a different status as work went on.
A nice feature with this online tool was that we could have a digital white-board to help us keep track over our tasks in our weekly sprints and have a burndown-chart so we could check how much we had done and how much was left in our current sprint.

Although it was very slow from time to time and lacked some features like a backlog, we still found it useful even though we could have updated our tasks more often. Having an online white-board can be real useful if meetings are hard to do in person e.g. groups are stationed at different locations, if we had to do it all over we would still used some similar program but faster and with more features.

Git

Working with git was a good way to be up to date in the code process when working together on the project. It was a bit tricky to understand in the beginning as we were all beginners and almost everyone had never used it before.

In previous projects we use Dropbox for syncing files. Compared to that experience we know that Git saved us a lot of time and problems. We could have been better in how we used branching and merging, and use more of the rebase functionality. We like the ability to use rollback and we will all be working with Git future projects.

Gitflow

We used the gitflow-model to help get a better structure of our git. The model was very useful as it made it easier to split the project into smaller parts for every feature so it could be worked on individually and then merged together. This also meant that we in some way used FDD (feature driven development) since we worked on each feature independent of the others.

Using gitflow worked well even though we did not follow the model to a 100% and the tree could get messy with alot of branches which could risk leading to branches being “rotten” and unsynced. f we had used the rebase function more we might could have gotten a little more clean tree structure.

For very large projects we think it might be more appropriate to use another technique since this could easily get cluttered with alot of different branches.

Working close together

We spent most of the time working together, around the same table. This way we could easily ask questions and help each other out. It was very convenient for us to work like this. Working close together also made it easy to make group decisions and arrange meetings on short notice. We felt that we worked better together than alone, most of the time.

Everyone was kind of new to working with Android and Git, but maybe working at the same place could slow down someone that is better than the rest. Working this way is probably often better for the entire group and project.

Working like this is something we would see ourselves doing in most future projects because it was more efficient than working on your own.

MVC-pattern

Using MVC-pattern in Android projects was a convenient way to build the app because Android uses XML-files as views and activities works as a controller and lastly our own classes is our models. By using the MVC-pattern makes the product more scalable and modular.

It takes time at first to implement this pattern but in the long run its worth it to reduce coupling and have a high cohesion, its highly recommended to use this if the application uses some kind of GUI to interact with the program.

What worked well

The collaboration worked very well and everyone in the group did their part and used the selected techniques. Even though we all had different week schedules we were able to have scrum meetings every other day and when we were working on the project we almost always sat in the same room.

Good planning

In the beginning of the project the user stories were too big and not precise. This led to some tasks taking longer time than planned. After a short period of time we learned to break down the task even further which made them easier to plan and work with.
Later on in the project the majority of tasks in the sprints finished on time and with a reasonable amount of effort.

Roles

We found it nice to have different roles that we could use to avoid conflicts and to make it time efficient. We had a product owner who was ultimately responsible of making design and feature decisions, he helped to quickly solve different problems like graphics and prioritize the features. We also had a scrum-master who was responsible of holding scrum-meetings and sprint-planning. Having one dedicated scrum-master lead to well organized and effective meetings.

What did not work well

In beginning we had too large sprints. It was hard to estimate how much time a user story would take to develop. Git were also hard to use and understand at the start and we had some problems with the .ignorefile in the first week when working on the project.
We felt that the lack of Android knowledge led to that even the smallest implementation took a long time to fully grasp. Likewise not developing a proper model and making class diagrams

from the start lead to bad development decisions. Although we believe it would have been hard to make good class diagrams when we did not know how to develop in Android.

Reflections over non process specific decisions

As having a Product owner it became naturally for the group members to ask him when taking design and feature decisions.

After a crash on the first private server we moved the back end to a commercial web service. In future projects we would think twice before using a private server without backup.

We chose the minimum API as low as API 8 to make the app accessible to as many people as possible. We had some minor problems to make our product work on older devices, but almost every time we found a work around after some research to solve the issue.

Group work related

As we described in the Scrum process we almost always worked at the same location, switching between a private group room and a public setting. This worked very well for the group as the change of environment made the work less linear. The private room was good for sprint-plannings and other meetings and the public space was better for raw coding.

We did not use pair-programming a lot so for the majority of the project everyone was working on their own tasks but since we were always working in group it was easy to help each other out when needed.

Everyone in the group participated with full dedication in the project so the workload could be distributed evenly.

Everyone in the group did not have the same schedule but we always found a decent amount of time to work on the project together.

Could have been done better

We feel that our daily scrum meetings were hard to schedule on a daily basis due to our schedules not matching. This led to that the group were often not fully assembled until after lunch. A lot of informal design decisions were made with the team not being fully assembled. In the future we will try to schedule our daily scrum better so that everyone in the team can be accounted for. We will also try to follow the scrum rules set within our project better.

Class diagram was something we would have liked to have done sooner than we did, to get a better overview of the project. The lack of diagrams made it hard to understand how the different features were related to another. Maybe the the features could have benefitted from each other if we would have had a clear view of the layout while implementing the application.

In the beginning of the project we should have taken time to fully get emerged into android programming before starting coding, not doing this led to several unnecessary problems down the line forcing code to be refactored.

We could had implemented the gitflow model even more as we did not use everything in it. We could have used Rebase to get a cleaner git and we should have branched and merged with more thought.

Feature branches became more like user branches in the end of the project, where a developer had a branch where he did all coding, not necessarily related to the feature. We believe that this was a result of when we in stressful situations did what we were most comfortable with and leaned back on old routines.

We did define the definition of done, but we were not the best to follow our instructions, that is also something we will keep in mind for future projects.