Post-mortem Report

Beike 2015

Authors

Adam Ingmansson
Alex Evert
Mikael Kajsjö
Simon Boij

Table of contents

Processes and practices

The agile method used in this project for our main development process was Scrum. We chose to use Scrum over other agile methods like Kanban as we felt more confident in using it because of lectures and the LEGO exercise. We also opted out of using pair programming as none of us worked the same hours and we did not have a good place to work at.

We decided not to use Test Driven Development as all of us had little experience in testing beforehand. The test were written after the code where finished as it was the easiest way of doing it.

We discussed using Google's Java Style Guide, but in practise we didn't check that our code met the standards because it took too much time. GitHub was chosen over other platforms, such as Bitbucket, because all of us had already used it in previous projects.

For documentation we decided to use Google Drive because it allowed us to write text in parallel, i.e. when we sit and write together we can all see what everyone writes. Everyone had prior experience in using Google Drive which made it a natural choice. For further collaboration assistance we used HipChat for internal communication using text, code and images.

We started by writing our user stories in a text file, but switched to Trello almost right away. This means that our user stories and backlog was written during the development process in Trello as Cards with attached tasks.

Scrum

We used a variation of Scrum with sprint length of one week. The reason we chose to have such fast sprints was that we felt that it better suited the project's short timespan. We still think it was a good decision because it gave us the opportunity to do more changes as they were needed. We also didn't have any stand up meetings, since we thought that it would be more of a hinder than help, as we didn't have a persistent workspace nor time schedule.

The sprint planning and user stories was really valuable, it gave us a clear view of what we wanted done each week and how to divide the project in manageable parts. Monday was naturally chosen as the day our sprint planning meeting, because it is the start of the sprint and week, to give us more of the week for development. This was later revealed to be a small problem as it collided with the monday lectures. Friday every week was chosen to have a meeting where we had our sprint retrospective and review. This was because we wanted to work from Monday to Friday and not weekends, which was not always followed.

We initially chose 20 as our velocity, where one point was one day per person. This was later changed to 40, with two points per day and person, because we halved the time each point was worth to give a better time estimation. The time estimates for the tasks worked

well overall and allowed us to distribute the tasks fairly even. Though, one problem was that it took us a lot of time to do them. Also some user stories were too big and should have been broken down to smaller segments. Another problem was the underestimation of some tasks in the beginning of the project, but the estimates got better in the later weeks.

Every week we changed Project Owner and Scrum Master to allow everyone to get the opportunity to try the roles out. But the roles felt redundant as we did not have a real stakeholder, we did all the planning of features together as a group and because we had good communication within the team. Therefore the main goal of the Scrum Master became to reserve study rooms, and of the Project Owner to bring "fika" on Fridays. In retrospect, we feel that rotating Scrum Master was not a good idea because you couldn’t follow up on the obstacles we discussed.

The first two retrospective was really good but we didn’t follow up the problems discussed as much as we should have, perhaps due to a rotating Scum Master. The latter meetings felt less useful, as the project would soon be over. The review was nearly non existent as we already had a good insight going into the meeting what the others had done. That we fact that we had very brief reviews was in hindsight negative as we needed a better understanding and constructive criticism of each other's code, i.e. peer review.

By using Scrum it was easy to change our idea as we realized that our original idea would not work because of latency in the Electricity API. One problem that we faced was that our task backlog nearly always was to short and each sprint we made up tasks for only one week. We felt that we could not afford to spend more time planning and needed to start completing tasks for the upcoming sprint. What we really needed was instead to discuss our goal and vision for the app. Perhaps if more time had been spent on the backlog or had a persistent Project Owner we could have been more efficient.

When we made the user stories we had some trouble thinking from the Project Owner's perspective. For most of the them we had the mindset of developers, rather than Project Owners. We looked at tasks that needed to be done and then making user stories that suited those tasks. In hindsight, we believe this was because we did not have a persistent Project Owner who was not a developer of the project. We think that the roles of Project Owner and Scrum Master would be useful for other projects while working for someone else, even though they did not really help us at this project.

Overall Scrum had a positive effect on the project. We would definitely use it again for software development for a company, but maybe not for a school project with rotating Project Owner and/or Scrum Master.

Common problems using an Agile process

By looking at the most common problems agile development teams have, which can be read on wikipedia[1], we see that there are some that we have had issues with.

Lack of overall project design

Now when the project is over we can look back and see that the design was lacking. We attribute this mainly to the lack of a fixed person as product owner, but also the lack of time attributed to planning over development. It felt forced to always continue and no one could say stop, and reflect over what needed to be done in the long run. Looking forward we see that we need to be better at taking the time to talk about the direction of the project. If we would have a external project owner this problem would have been mitigated by the fact that they don't have any connection to the develop process.

Product owner role is not properly filled

As everyone had their own idea of the app and there was no one person that decided, the role was not filled and instead everyone came up with their own solutions. The problem lead to the backlog being short, user stories was written from a developer's perspective and the goal/vision of the project was never clearly decided. The decision that everyone should be content with all design choices led to the project never having a clear goal and vision. As earlier stated an external project owner would have been a better solution.

Problem-solving in the sprint planning

A common problem during our sprint planning was the derailing of the sprint planning into a discussion about design and how a particular problem should be solved. It dragged out the planning session and we did not have time to fill the backlog properly. A Scrum Master should have been tasked with keeping the meeting on track.

Scrum master and Project Owner as a contributor

We let the Scrum master and Project Owner have the same velocity as the other members. Which lead to that the roles wasn't effectively used, as it did not feel like a real task to act them. The solution to this would be to make the roles as tasks or lower their velocity, even at the cost of getting less development time.

Allowing technical debt to build up

The project was short and this means that it never became a big problem. But if we were to continue we would need to refactor and change the structure as it would grow fast into a project that is difficult to manage. A large part of the problem was that the Definition of Done was not followed, which resulted in tasks that were almost finished but not quite. If the Definition of Done would have been followed and assigning someone to facilitate any merging would have greatly helped to mitigate this issue.

1 https://en.wikipedia.org/wiki/Agile_software_development#Common_agile_pitfalls

Git branching model

We tried to follow Vincent Driessen's branching model[2] and in general that worked quite well for us. The project never felt cramped and we mostly avoided undoing each other's work. Integrating new features to the develop branch also worked well but it's hard to tell what part of that that is due to our branching model and what came from the task assignments. We thought about setting up some automated building and testing but it never felt like a good time investment. The same goes for automatic deployment of our server.

On a few occasions we had many developers on the same task and on the same branch and our git log isn't the cleanest but we think it worked out fine. Apart from the purpose of the branches we did not really discuss "coding style" for how we should use Git the way that we did for the Java code and some code have been committed in way too many commits but we had no real tangle at least.

At first we thought about using Git for our documentation, but it has some of its major weakness as it is not suited for PDF documents and other non-text formats.

In conclusion this branching model works well and has clear advantages over for example a single branch approach.

Other tools and technologies

Trello

We opted to use Trello as our task board because it seemed easy to use, and had the functionality that we needed, in other words to help us to keep track of each other's progress and not to mix up assigned tasks. Trello have worked well and we never felt the need to find another task management software.

Web server

We wanted to have a shared high score system amongst all players and this led us to setting up and using a web server. In hindsight, we feel that this was unnecessary in the scope of the project. We should have opted for a simpler local score keeping, since it was not that important to have a shared score board. We initially was very focused on the multiplayer aspect of the game, but later changed it to be more of a single player experience. This meant that the importance of a shared score diminished. This was a significant portion of focus that could have been used in other areas of the project.

[2] http://nvie.com/posts/a-successful-git-branching-model/

HipChat

We started using Facebook as our main communication tool, but quickly switched to HipChat. This was mainly due to Facebook was not suited for code related conversations. HipChat worked very well and the notification system made sure that people could talk specifically to one person easily and paste formatted code or images. We set up different rooms to facilitate communication to / from the Scrum Master and Project owner, but this was rarely used.

ElectriCity Innovation Platform

If we would have a choice, we probably would not have used this API. We felt that the format that the data was delivered in didn't match our idea and thus presented us with problems in the development process. The latency of the API lead to us having to rethink the whole idea into a different direction. During the project we always felt we were working against it, not with it.

Google Drive

When we started writing the first documentation it was quickly apparent that Google Drive was a very good fit for us as it allowed us to simultaneously write in one document. Later it was realised that this fractured the contents of the project. Which made it harder to keep track of what we had done and whom had done what. We still think that it was a good idea to use Google Drive, but we should have made continuous updates into the Git repository. In addition we should have been adamant in tagging the sprints and committing the retrospectives but between Trello, Google Drive, Hipchat and Git. We had one too many tools to think of that.

Time distribution

In total we have spent around 500 man hours (mh), one hour for one man, on the project and we generally feel that more time would not have ended in a better product. What we needed was a different focus, not more time. We spent about 200mh before the project itself started. This time includes lectures, but also pre-planning such as idea to concept and the vision. When the project started we spent about 20mh per week on sprint planning and time estimation, where we had one meeting on mondays and one on fridays. We planned about 13mh per person each week for development, leaving 2mh as a buffer for unforeseen events.

The following table shows the approximate time spent per week by each team member during the development:

Member	**Man hours**
Alex	20
Adam	15
Simon	18
Mikael	15
Total	68

We estimate that we have spent about 25 mh on various kinds of documentation. For ElectriCity Innovation Challenge related tasks, such as writing up concept document and making the video, we each spent about 25mh.

Team dynamic

We aimed to have a open and constructive team, where everyone could say what they feel, without being judged. This has been very important when we discussed different parts of the development process and what we want the app to look like as a finish product. We often had different views and opinions, but always managed to compromise. This lead to a solution that everyone was satisfied with, but not necessarily thought was perfect. This took a lot of time, but in the end was a very good and rewarding process. We really felt that all team members were active in discussions and contributed equally to the project as a whole.

We felt that we had good communication during development, but faltered in the aspect of collaborating when tasks was completed and should be integrated with the rest of the application. To mitigate this problem we should have made a proper version, with a built and signed apk along with a tagged commit, so that every sprint would had a definitive end. We also felt we had problems communicating when you were blocked and needed someone else to finish their code.

We have solved smaller technical problems together on Hipchat, but never took the time to plan and discuss the overall design and parts of the project. This contributed a lot to the lack of direction of the project.

Our meeting had the tendency to be prolonged. It was easy to discuss technical issues i.e. when the meeting was to plan the sprint. During the meeting sessions we sometimes got sidetracked with more personal discussions. This was maybe due to the fact that we did not keep meeting minutes and the Scrum Master did not keep the team on track.

We all have different preferences on working hours and decided that each person had the responsibility to plan and do their task(s). All the while everyone had an eye on the chat and was easy to contact if you needed to ask a question or some help. It was very comfortable to work from home and the hours of one's choosing, but this probably lead to less collaboration on the project's direction. On the other hand at the same location could also have lead to us getting distracted by less relevant discussions.

ElectriCity Innovation Challenge

The interest in the competition was very low and participation felt forced. It didn't feel applicable to our situation with making an Android app and as such the later workshops didn't feel constructive to spend time on. In addition we felt that the deadline for the concept was too close to the workshop and had already finished by the time it started and thus we didn't attend. The whole thing felt like a marketing strategy and since their api is in a alpha state it will not be possible for us to look back and try our app in a few months.

At the start we selected the representative by asking who wanted it and by group consensus. All communication from stakeholders was forwarded to all members and discussed on meetings and/or hipchat. However, we felt that a meeting with the stakeholders during the project would have been very productive.

Conclusions

We feel that the biggest problem we had was at the start when we had two very different competing ideas. This lead us to spend many hours trying to decide which idea we wanted to realize, instead of actually start expanding the concept and clarifying it. One reason for us not to decide on one idea was that the lack of information made us worried that the EIC API was not up to par with what we wanted to do, which later turned out to be true. The final decision was not made until the last week before the first deadline of the competition. Some of the work we did before the first deadline was later removed as it was not relevant to our project, but we learned a lot about android development that we took with us. We feel that this could have been solved by being less democratic when deciding which idea to go with. Another way this problem would probably been less severe if we had been an odd number of people, then there would have been a majority in the democratic vote.

We started the development process with solving the problem using our prior experiences with ordinary Java solutions. We came to a point where we could not continue writing our code in this way, and had to rewrite some of our solutions the Android way which took extra time. This might have been difficult to avoid since we are new to Android.

In the project we made the assumption that we, as a self organizing team, should not ask too many questions. In hindsight we shouldn't have made this assumption and asked questions to both mr. Burden and EIC board as an external feedback. It would have helped us as a team to have a halfway review about how we are doing.

Without a clear vision we had trouble prioritizing what was the most important part of the it. What we ended up doing was the complete opposite and making most of the peripheral functionalities first. We never clearly decided how the game would look like as a finished project, we just pushed the decision forward. We started with the simple functionality that we were sure that we could use in the project, because we felt that we needed to get started and we did not want to work in the wrong direction. What we should have done was to work on the core aspects of the app and after that work from that foundation, since it is irrelevant if we have peripherals to a non functioning game. We think that the end result is, in spite problems, a fun and engaging game.