Post Mortem Report

Group NULL:
Elvira Jonsson
Ivar Josefsson
Francine Mäkelä
André Samuelsson
Malin Thelin
Joakim Thorén

Chalmers University of Technology
Software Engineering Project (DAT255)
2014-10-30

1. Introduction

PickUApp is an android-app for matchmaking drivers with hitchhikers. The driver gets to set his destination and the hitchhiker gets to set where he wants to go. PickUApp then matches drivers with hitchhikers, informing them of each other.

The purpose of this project has been to try out agile development - mainly scrum. To learn to program on android and other new technologies (such as AGA and Gradle) was also a goal.

This report will elaborate on developing with scrum, developing with android, application architecture, project result and what could've been done better and what was successfully accomplished. It will discuss the advantages and disadvantages of the different developing processes and techniques for this project and analyze in which situation(s) each process/technique would be appropriate to use, based on our experiences. It is not an objective report, it is subjective.

2. Developing with Scrum

2.1 User Stories

The user stories was created during the beginning of the project. No user stories were created later on. The user stories were probably too large and could've been narrowed down. As a consequence we only had three sprints. One setup sprint, another week long sprint and another three week long. The three week sprint covered two large user stories. These should've been viewed as epics instead of user stories, which would redirect focus from the epics to actual small user stories and enabling smaller sprints.

Perhaps user stories was just overhead and didn't really contribute towards any understanding of PickUApp or what work that should be carried out. When deciding on what to do each week an oral agreement worked just fine. Every group member had sufficient overview of our collective goal. This is probably not the case for larger projects.

2.2 Product owner

Joakim Thorén had the role as product owner. The product owner was responsible for creating user stories and presenting them during sprint meetings. This was done during second and third sprint. As elaborated on in User Stories, the third sprint was three weeks long due to large user stories. Since these two user stories were so large, it was said during sprint meetings that we would continue the previous sprint in order to finish the two user stories. The product owner was also the meeting administrator during sprint meetings. During each sprint meeting the product owner proposed what features that should be done by the end of the week. These features should also have been documented user stories if following the scrum-methodology. As for this project that might would've been project over management. If we would work full-time on this project this would probably not be the case.

2.3 Sprint meetings

On Mondays there was a sprint meeting where Joakim (the product owner) presented a suggestion on which User Stories to implement during the following week. The advantages of having sprint meetings was that all group members got an idea of what needed to be done. Since the whole group was gathered the meeting resulted in a group programming afterwards. The downsides with the sprint meetings was when there was too much workload on a single sprint with resulted in sprint meetings that weren't as effective since no new sprint was started.

2.4 Scrum points

We decided to tryout setting scrum points but the method was misinterpreted and points was divided amongst user stories and not the tasks. A realization was also made that because this would be such a short project the task of dividing points would just be another overhead thing we would need to do, after this it was decided not to use scrum points. As a consequence we couldn't use a burn-down chart. But we can see the point of it in a project that stretches over a longer period were meetings would be held more regularly and especially where you would have a target customer.

2.5 Backlog

There were two ways of creating tasks to put in the backlog. The first, and most obvious, was by have a look on the user story selected by the product owner, and figure out what had to be done in order to fulfill it. Also, each group member had the responsibility to, whenever they found a bug or something that had to be done, put it in the backlog as a task. Since the person who added a task necessarily wasn't the same as the one who solved it; all the tasks had to be short and well described.

The backlog gave a good overview of what had to be done and if the project was moving in the right direction. This made it easy for each group member to pick a task, and therefore keep his or her workload on a reasonable level and at the same time no one had to worry about what to do next because there were always tasks in the backlog. It also gave all of the members a chance to work with whatever they found interesting and wanted to learn more about.
The only disadvantages about the backlog was that it actually had to be used. All group members were efficient in assigning tasks to themselves, but not always was a new task added when needed. This probably come out from the fact that no one in the group had worked with backlog before and therefore wasn't used to put the problem away for a while and maybe let someone else solve the problem.

2.6 "Stand up" meetings - Scrum meetings

We did do scrum meeting once a week and we feel that it was useful, they did not take a lot of time and ensured that everyone working on the project was roughly on the same page and had an idea of what others were working on.

One of the biggest perks of these meetings was that we made sure no one got stuck over a longer period of time on one specific issue, we also really liked the efficiency and quickness of the meetings and can't see any negative aspects.

2.7 Scrum master

André Samuelsson took upon himself the workings of the scrum master.
The scrum masters only task was to administer the scrum meetings. No other scrum related task was really relevant to the project. It was a nice role to have because there was no doubt about how these meetings would be handled.

2.8 Time spent on scrum processes

The group spent approximately 70 hours on scrum processes in total for the entire project. What we spent individually is summarized below.

Individually
Joakim Thorén: avg 3h/w
André Samuelsson: avg 2h/w
Ivar Josefsson: avg 2h/w
Elvira Jonsson: avg 1,5h/w
Francine Mäkelä: avg 1,25h/w
Malin Thelin: avg 1,25/w

2.9 In which situations would we use Scrum again?

On large projects (larger than 2 months, or full-time projects). Doing agile development with backlog and sprints is probably a very good idea no matter the time frame of the project.
Or on project where meetings are held more regularly. i.e at least 4 meetings per week

2.10 In which situations wouldn't we?

When doing software-projects in school. Some aspects of scrum increases productivity and improves group-management, but applying 100% scrum, such as burn-down-chart or scrum points seems to be project over-management. Scrum doesn't feel like a methodology that is scaled for a two-month school-project. Scrum comes with some bureaucracy that seemingly adds more overhead than it adds value to the project.

3. Writing the code

Every group member has been developing a little in most parts of the application. The great advantage has been that everyone has been able to try android and received an understanding of how the application works. This has been good for personal development, but maybe not as time effective as it would have been if each member have had a specific field of responsibility.

3.1 Version Control

Git has been used for version control and all group members has been working towards the master branch at all times. The reason is that we wanted to avoid any large merge conflicts which we did. We briefly considered developing on several branches but it didn't seem necessary for this project. Having one branch has been extremely time effective.

3.2 Gradle

Given the time and value gained from Gradle we probably shouldn't have used it. While it worked excellently from terminal when it was correctly setup, tests didn't work when running from an IDE. We spent ~15h/person on setting up gradle. That is approximately 90 work-hours. It would probably be more time-efficient to download and setup dependencies and tests manually. The high amount of work on gradle is because nobody used it before, so naturally it took longer than it would with gradle-experience.

3.3 Test Driven Development

Part of the code has been tested through the development. One advantage of having tests is the ability to check if the code does what it is supposed to. Another (perhaps greater) advantage is that you get fast feedback on whether or not changes in the code affects other parts of the application, which have been useful in the project a couple of times. It is also an advantage since the tester must consider and validate the purpose of the class.

The disadvantages, on the other hand, was that setup of the test environment was technically difficult since the code that was tested was completely separated from Android. JUnit 4 was used and the tests could only run when using gradle from the command line, which was problematic for group members who was using an IDE. This caused build errors being published to the repository at times. Another problem was that the Jacoco plugin didn't work with gradle, which made it very hard to know if the tests covered all of the code in the model (this is still unknown). The functionality of the model is complicated to understand, which made it hard to test.

Despite the time it took to setup and get all group members to build the project from the command line and write the tests, the usage of test driven development was time efficient. Bugs were found that probably could have been unnoticed or hard to find, which saved a lot of time. The bugs were easy to solve thanks to the tests.

3.4 Group programming

The group programming session (usually after meetings on Monday and Thursday) were good because of the opportunity to solve any larger problems and help out group members that needed it. The group programming was a great chance to discuss possible solutions to specific problems. The group sessions has contributed to a better understanding of the entire program for some of the group members. This was achieved by listening to the discussions and having someone explain certain parts of the code. The disadvantage was mostly connected to planning. Some group members has long travel time and lost time that could have been spent working on the project. This has still been the most time effective form to program.

3.5 Pair programming

We practiced some pair programming and found it to be quite helpful, if both programmers are on the same page you can feel relatively sure that work is progressing in the right direction, there is also the perk of having to explain your train of thought incase one person does not understand your idea, thus helping both programmers to find problems and issues with the planned implementation.

Pair programming was also helpful in us keeping a high efficiency, it was easier to stay focused and continue programming when you were two programmers working together. We also found that pair programming was really effective when solving the more complicated bugs. It resembled rubber duck debugging on occasion.

However, while there are several benefits with pair-programming it also consumes twice as much workforce. It’s not clear if it is twice as productive.

3.6 Individual programming

Individual programming was carried out at times when group members knew what should be done and were able to do it themselves. Using backlogs with waffle.io was a great help with individual programming as you didn’t need to discuss with other group members beforehand what to do and what not to do. Without backlogs individual programming would be much harder to carry out efficiently and there would probably be more duplicated work since you couldn’t easily check what others were up to.

3.7 Time Spent on programming

The group spent approximately 425 hours on programming in total throughout the project. What was spent individually is listed below.

Individually
Joakim: ~17h / week (avg 4h/meeting x2meetings, avg ~9h/home)
André Samuelsson: ~16h / week(~8h from meetings + ~8h home)
Elvira Jonsson: ~14h / week
Ivar Josefsson: ~14h / week(~8h from meetings + ~6h home)
Francine Mäkelä: ~12h / week
Malin Thelin: ~12h/week

4. Technical practices

4.1 Application Architecture

The architecture was directly derived from the application flowchart. After making our vision we created a flowchart to further understand our collective idea of PickUApp. At this point it seemed reasonable to model the flowchart instead of traditional object-oriented programming since it was fairly straightforward what classes there should be and what their purpose would be, as opposed to defining classes from subjectives of a descriptive text. This essentially became a state-machine with sequences who contained onEvent-methods which handled relevant events. It later on became obvious that events was actually exits from steps in the flowchart. It became very easy to understand what classes was needed and what purpose they had - creating tasks in the backlog wasn't hard at all and discussion about intention of backlogs was very seldom required.

Another approach to modeling the application would probably not make things as obvious as they are now, in that manner this architecture seemed like a good choice. On the other hand there were little to no experience in the group with this type of architecture so many new and unexpected questions arose. Passing data from a sequence to another, how the view should change when a sequence changed and how events should be handled were some topics that caused confusion and took a lot of time. All of these problems were eventually solved, but it introduced some overhead and complexity which was unexpected. It unfortunately wasn't within our timeframe to reiterate and try to find the most optimal and perfect solution to these problems.

Passing data was solved by using an singleton Eventbus, which simplified communication via view and model. Without the Eventbus this project would probably be nowhere as finished as it is today.

As for separating the model from android, it gave us the advantage of being able to start coding on the model in parallel with learning android. If each sequence would be an activity on it's own, the work on the model would be slower and delayed. While not required by this project, but a benefit nevertheless, separating the model from android makes the program easier to port to other platforms.

5. Reflection

5.1 What worked well

The backlog worked excellent due to making it much more easy to organize and distribute work across the group.

The time we spent programming after the group meeting were our most productive. It great to be able to speak to people directly to resolve issues.
When we compare group programming to pair programming the biggest difference would be the code quality. In most cases the code is more understandable when written and overviewed by two persons instead of one.
In aspect to these two we have individual programming which is good because you did not need to synchronize with other team members and you could easily take tasks from backlog and work on by yourself.
So all of these ways of programming work well and makes the group able to work dynamically.

Version control has been working well mostly thanks to the fact everyone was familiar with Git before the project started and that everyone pushed to master all the time and in that way avoided larger merge conflicts.

5.2 What didn't work well

The project was slow started, the biggest cause of this was the time spent on getting AGA (Automotive Grade Android) to work. Essentially the first three weeks was spent entirely on figuring out what AGA did and how to use it. Setting up the tools was not trivial and documentation was lacking.

We applied scrum to our project much later than we should due to receiving the lecture on scrum too late. Ideally that lecture would've been amongst the first three lectures, but it was the seventh. The course gave the general impression that AGA was the only thing that mattered, which caused us to work to much on setting up AGA. In the end we didn't use AGA anyways, not because we didn't know how to, but because none of it's functionality was needed in PickUApp. Especially not when considering the value-time-ratio. We should've focused on scrum directly and ignore AGA, but this didn't seem like an option at the beginning of the course. We had to redo all our processes and how we earlier decided to work when scrum was presented to us. This wouldn't have to be the case if lecture on scrum was earlier.

Many of the scrum processes have worked well. Since the project has run on half time it hasn't been possible for more standup-meetings, due to the fact that there hasn't been that many changes to the code from one day to another. The standup-meetings were held once a week which was enough. The feeling of collecting the group once a day and starting up together was unfortunately lost because of the pace and scheduling.

5.3 Group dynamics

There has been a general positive attitude on meetings and when programming in group. People have been listening to each other and helping each other out when needed. There has been different levels of technical knowledge amongst the group members which hasn't been a problem due to the openness of the group. A lack of knowledge hasn't been something to be ashamed of and help has been offered to those who ask for it.

There has been conflicts in the group mostly regarding the architecture of the program and project management. The conflicts in the group have been solved through discussions. Someone has suggested a different way to do something and eventually got the approval from the rest of the group. This has been time consuming but on the other hand there probably is not another way to resolve things that would've worked for us. During the first half of the project the amount of work carried out wasn't as high as necessary for reaching the goals we collectively set at the start of the project. This made the workload on the sprints larger. In the end the original vision of the final program was compromised, which created frustration amongst some group members.

5.4 What should be done differently next project?

- Make sure the development method is final (don't want to change methodology to scrum mid-project)
- Analyse if proposed API's are really necessary (IE AGA). Add and learn API's when the project requires it. Don't add and learn API's if the project might require it later on.
- Appoint someone who is responsible for dev-environment which makes sure everyone uses same version where needed etc.
- We should have a formal written, extensive, definition of done instead of having an oral one
- We should have designated someone to design PickUApp graphically
- There was uneven workload on creating backlogs which could have been distributed more evenly
- We should have a written formal definition detailing the purpose and size of a backlog-task