# Software Engineering Project

## Post-Mortem Report

Grupp 9


| Name: | e-mail: |
| --- | --- |
| Adam Grönberg | gadam@student.chalmers.se |
| Jonas Lindberg | jonlin@student.chalmers.se |
| Azer Vilic | azer@student.chalmers.se |
| Magnus Dannerstedt | dannerst@student.chalmers.se |
| Niklas Helgegren | nikhel@student.chalmers.se |
| Elin Karlsson | elikar@student.chalmers.se |
| Nora Idbratt Lundgren | idbratt@student.chalmers.se |

## Processes and practices

We have worked using parts of the scrum process such as sprints and user stories. In addition we used pair programming, especially in the beginning. The advantages of using scrum in our group was that we followed our goals that we set up each week thus making our project move forward at an even and efficient pace without overworking the group. However, we did not have daily meetings in our group. This was simply due to that our schedules did not match. Four of us had another project going on simultaneously and the remaining three members had another course all together to keep track of. This also forced us to split the work into two longer days where the whole group sat together and worked. However, we did have a short meeting on Tuesdays before we started to work where we brought up issues that we had and what needed to be done for the week.

We have worked for a total of 6 weeks and spent approximately 20 hours each per week on the project for a total of 840 hours.

## Scrum

Scrum was quite effective for us mainly because we followed the sprints strictly making our work highly efficient. It helped us visualize and give us a fair understanding of how far we were in the project.

Scrum is intended for full time work at one location. Daily meetings don't work well without that premise. It also assumes there is some sort of base of operations with a whiteboard or equivalent to use as a scrum board.

We would use scrum in future projects if our schedules matched better so we could have daily meetings and spread our total work time over the week better.

The benefits of using scrum compared to a more plan driven process were that it was easier to add and remove features, which made it easier for us to see how the work proceeded and easier to split the work between the group members. Scrum was way faster and more efficient, had we not used scrum it would have taken longer to get started because of all the documentation it would have involved. We would probably spend more time on connecting the different pieces of our project if we used the waterfall technique. Because having the ability to just work on a special minor part and then merge it whit the big part was a smooth way for us to work all of us at the same time.

We believe that scrum did provide us with a good way of working for dividing the work between the group members, so that the most of the time all members felt that they did have something to do. Doing a sprint review after each week allowed us to better plan for the next week what we would implement. It was also effective that if we realized something would take too long time we could scratch that and choose something else to work on instead. It did help us to see how we proceed as well and we felt that it was encouraging to

see the progress we made each week. It is positive that you are allowed to change requirements during the process compared to a waterfall process where you must decide all requirements from the beginning.

Drawbacks compared to waterfall.  There were problems implementing testing and keeping it up to date. We felt that creating a test case and keeping it up to date during the entire remaining project did take some of the resources that we could have used for improving our or completing other parts of the project.

In waterfall all the requirements are known from the beginning so you know exactly what you have to focus on compared to scrum where the requirements may and certainly will change during the process which is a drawback.

## Pair programming

In the beginning of our project we worked a lot using pair programming since the problems were more complicated and bigger. However during the end of the project we decreased the amount of pair programming and worked more independently since the problems became more divided and smaller. We still used pair programming to some extent during the end of the project when for instance one person bumped into a difficult problem.

The advantage of using pair programming was that we solved the difficult problems a lot faster than if one person was given a complicated task to solve by them self. This in return made our progress faster.  A disadvantage was that there were sometimes two wills that wanted to implement the code in maybe slightly different ways. However this was not a common recurring issue.

We would use pair programming in a future project as stated if we would bump into difficult problems in the project. We would not use it if all of the problems were small and much divided.

Our teamwork worked very well and we split up the tasks in a good way without any arguments. The pair programming was also effective and made our more complicated problems solved faster.

In the beginning we tried to use the Android SDK Emulator, but because of recurring problems with delay we decided to run the app on our phones instead. Not all members in our group had an Android phone, but this didn't cause much trouble since we worked together in pairs most of the time.

## LibGDX & Git

We used LibGDX when developing our app. LibGDX was proposed to us by a teaching assistant during our first meeting. We used it to draw squares on the screen which means it handled all the graphics for us. It handles the main loop and life cycle of our app. All the

sound in the game like the music and bomb explosions is handled by LibGDX. It was very smooth using LibGDX.

If we were to do another project we would devote more time to learn the tools, such as LibGDX and git, from the start. Much time in the beginning of our project was wasted because we tried to go directly to coding before we knew enough about libgdx and the available tools there. Many of the guides about libgdx we found turned out to be outdated. We would have known this if we had spent more time exploring the official API.

Git took some time to learn as well. If we had taken a few hours to learn it together we could have avoided downtime later when we waited for eachothers merge/sync to avoid complications.

## Known issues

- Menu buttons don´t work if you press the lower part.
  We chose to not fix it since it isn't noticeable when you run on android unless you know about it.

- Testing won´t work when the animation is active so we have to comment the row which calls animation while testing. We have been working a lot on it but haven't found any solution.
  We thought about using a variable that determine if animations should be used but decided that it would be bad programming. Since it´s a game we focused on testing most parts manually by playing repeatedly and trying to make the game crash.

- When you got the last upgrade on your laser (quad) it increases the cost to a unreachable amount(something like 100 000 000 ). Its not a bug just a design choose to mess with your head.

## Minimum requirements

We decided to set API requirements to release 14 (android 4.0). This was the lowest API that would run our code.

For hardware requirement we set Samsung Galaxy Note GT-N7000 equivalent and later. This is the oldest phone we have that could run the app smoothly.