# Discbuss - Post Mortem Report

**Authors:**
**Oskar Willman**
**Oskar Rutqvist**
**Jakob Holmgren**
**Oscar Beronius**

## Processes and process decisions

### *Scrum inspired developing*

We started out our project after a scrum workshop early on to start developing material to use a regular scrum process. The first thing we did was dividing our project into smaller pieces as well as rating them by workload and relevance. We decided what we needed to get done the first week when the project started and began working. A week from that point we realized that the regular scrum process wouldn't work very well for our team and decided to stop using the regular scrum. We still used an approach to the development that was rather similar to scrum however. We used a workboard called "*trello*" that held cards of tasks that needed to get done and each individual could assign someone to that card. The trello board was updated by the respective group member when the task assigned to them was complete. They could then pick another card to work with and so on. Once we felt that the board had too few tasks we had a meeting, spoke about what we had done, what needed to be done and set up new cards to work with.

Please note, when we refer to Scrum in this report, we'll refer to our approach to it and the way we used it, which as stated above is not the real or complete scrum experience.

### *Standup meetings*

As we started up with a more scrum-like approach we felt that standup meetings was a good plan. The first week as a developing team we held three standup meetings, second week we had one early on and then we decided that this wouldn't work well with our new work process.

### *Pair programming*

We did some pair programming, especially in connection with the meetings and when we had difficulties. When we got stuck and had problems getting forward we would ask someone come help out with that issue and sit down together to solve it.

### *Git flow*

We used a well-known git model called git flow in order to keep a good branch structure, which resulted in us having less merge conflicts than one might have if a model like this would not have been followed. Given that everyone was aware of the model before, we didn't need to settle on "rules" about how and when to merge between branches, it sorted

itself out. Further reading:
https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

## Our project, step by step

### *Research & Startup*

The first two weeks we mainly attended lectures, started doing some research on android development and tried figuring out what project to do. We all attended the workshop on scrum to. Alongside with these activities we discuss what to do in our project. We all put in about the same amount of effort, about 15 hours week one and the same amount the second week.

### *Project week 3*

Week three was dedicated to developing our concept. Oscar Beronius and Jakob Holmgren spent a lot of their time coming up with ideas and refining the concept, as well as finishing it during a workshop(konceptstugan). They spent about 30 hours each on this during week 3.

Oskar Willman also attended both days at konceptstugan. He did not give much input during the first day, but spent a lot of time during the second day in order to finish up the documents as well as the YouTube-video, all of which had a deadline on that day. This adds up to about a little under 10 hours of value time.

Oskar Rutqvist didn't attend any of the days of this workshop but finished of a little work with the concept report. But this only resulted in a few hours of work this week.

### *Early development phase*

Our plan was to start developing the application early in week four, but at the start of the week we still had much work left before we could start coding. We had to decide what approach to use for the server and database. We mainly focused on socket.io and node.js alongside with firebase. We decided to use firebase after lots of discussion, and soon the git repository with project structure and things were up and running. We had some different areas we were focusing on, three of us had some experience in android development from earlier and could focus more on the general project. The total effort put in this week was 10 hours per person, except for Jakob with his 15 hours and Oskar Rs' with his 20 hours.

### *Project week 5*

Once the structure was up we could finally start coding. The basic functionality was created by Oscar B and this took about 10 hours. Then he helped Joakim building the chatroom and spent another 15 hours this week on that. Joakim put in a total of 25 hours this week on Chat functionality and a login function once the chat was running. Jakob spent this week on creating a user registration (firebase) for our application. He also tried finding BSSIDs' for the busses and ended up contacting Keolis. This is a total of 25 hours spent on the project this week.

## *Project week 6*

This week we had user registration, and a chatroom working within the application. Oscar B spent the main part of this week on fixing the chatroom with both functionality and design for a total of 25 hours. Jakob also put some effort into the chat functionality and came up with the idea to create the ChatAdapter and he did efficient work for 15 hours this week. Alongside with this Joakim kept developing the login function to create a auto login function so that you wouldn't have to type your login information every time you started the application. He completed this functionality and added a login screen. 25 hours week 6 for Joakim. Oskar R worked on functionality to show "my profile" and functionality to edit the displayname for a user. This together with his work on changing password functionality meant a total of 15 hours spent this week. Oskar W was working on minor things and ended up only spending about 3 hours on the project this week.

## *Mid development phase, producing the actual app*

Joakim extended chat functionality to include the possibility to chat privately with a user as well as improving the general chatrooms. This meant refactoring a lot of the code handling the chat. He spent almost 50 hours this week. Oscar B also spent almost 50 hours this week refactoring a large portion of the other code in the project and improving the functionality of connecting to the buses. He also found some time to improve the chatroom design further. Oskar R worked with the action bars in the application this week and put in 15 hours of work on this. Oskar W created functionality for logout and completed the registration functionality and spent 15 hours in total.  Jakob worked on functionality to show the amount of users active in the chatroom and this included adding them when they started the chat room and removing them when they closed the chat room. The drawables in the application Jakob created this week as well. 20 hours Jakob spent working on this functionality and design.

## *GUI enhancement and bug fixing*

All members participated at the competition final and preparing for this which meant everyone worked for 20hours on this project this week. Jakob focused on some bug and interface fixes, Joakim and Oskar R worked on completing design and functionality of the actionbar. Oskar also fixed some general design in the application and Joakim spent some time creating the functionality to remove chats between two persons. Oscar B finished up work from the week before and removed some bugs with the buss connections.Oskar W improved the profile activities and did some bug fixing.

## *Post mortem report and cleanup*

The last week we have mainly focused on writing the last documentation, post mortem report and cleaning up the code. Oskar R and Oskar W focused on the report and other documentation. Oskar R has put in 15 effective work hours this week and Oskar W has 20 effective work hours this week.

Oscar B and Joakim S worked together to refactor the usage of action bars in the application to come up with an optimal solution. They managed to complete the refactoring in 30 hours spent each.

## Process decisions

### *Pair programming*

Pair programming made it easier to get a good understanding of more aspects of the project because one person who had worked more than others on a specific part could introduce others to it. We thought this was a good learning practice, for both parts, both learning to read someone else's code quickly as well as learning how to quickly summarize what you have done.
Pair programming was also a good method for us to solve certain problems. This was because we could discuss our ideas and exchange our thoughts with each other and thereby gain some new insights on how to solve the problem. The problems could be a lot of different things, decisions on how to structure the code or solving some major problems within the application or even something much smaller, like inspecting the naming of methods.

There are also downsides to pair programming. One of the hardest things for us was spending a lot of time together. The reason for this was that we were reading a lot of different courses and being busy on different times outside of the regular work hours. That means that we worked with pair programming when we felt that it was very useful, for solving a major issue or inspecting someone else's work when they asked.

### *Standup/Smaller meetings*

Standup meetings is really helpful when working in a project like this. But It does not always work very well. The fact that we all read different courses made it hard for us to get everyone together at the same time. We changed this into smaller meetings where the ones who were able could attend and update each other and the workboard on what needed to be done and what had been done since last time. After that we sat down and started working together. This worked very well for us because of our different levels of coding experience and android knowledge.
This worked well for our team since we didn't have anyone acting as project manager which led to meeting agendas and structures was missing when we tried to sit down on more "regular" meetings. We also think that one of the main reason this worked this well for us was that the teams are quite small and we are not more than five persons in the group. If we would have been a bigger group the more structured meetings with agendas would have been a better solution. That would also have been the case if we would have use a scrum process more similar to the "regular" one. That requires better structure and management than what we had during our project.

### *Scrum inspired development*

Our scrum experience was a bit different from how we believe it's intended. Mainly due to the fact that we spent a lot of time working together and in small meetings. This kept us very

up-to-date on what was happening and ensured we were all headed in the same direction. We believe this made our scrum process very efficient once we had a scrum-structure working for us. In terms of keeping track of the current state of development we used a website called Trello and set up sprint-boards (more on this later on).
One difficulty with scrum was getting started we believe. Setting up the cards and planning the sprints take quite some time and given that we take many different courses and our schedules differ a lot because of it we had some problems finding the time to gather up for meetings like this. On top of that before we could get started learning scrum properly (in excess of lectures) we needed to learn how to make a android-project, study the electricity challenge and come up with great project ideas.
We ended up starting out a bit late with our sprinting, during week four of the course. But once we got started we set up some cards we could work with and decided on how we could go on from where we were. A week and a half from that point we changed our process from regular scrum to what we thought worked better for us. We started updating the work board on trello more and started working more together during several hours of the day. The ones working updated the board and this way the ones who couldn't attend would not miss out on what had been done.

### *Git flow*

During the full course of the project we used git to handle project versions effectively. When developing the application, each and every group member would either have their own personal branch for developing, or create branches for specific application components. When a component was finished, the main branch was merged into the personal branch in order to solve any merge conflicts, and then pushed to the main branch. This allowed the group members to develop application components remotely, without having to worry about any fatal merge conflicts besmirching the main development branch. The git flow worked flawlessly throughout the project, and is something the group members definitely will use in future projects.

## Project working process assessment

During our project we felt that our pair programming worked especially well. The cooperation between the members of our team has worked extremely well, and solving problems, issues and problems internally within the group once they came up has worked very nicely.
We had trust in each other's effort. Despite the fact it took some time to get the full idea ready, once it was ready, we had it well-defined enough in combination with everyone agreeing upon the end destination and our way there. This made sure to minimize unnecessary complications by having everyone focus on the same goal and made design/functionality decisions easier to set.

We had problems especially in the early phases getting everyone to attend meetings and just get going. Mainly due to poor communication in combination with different schedules and, at least to begin with, bad work morale. We had some different levels of ambition which became a bigger problem than necessary because we didn't have anyone who wanted to take responsibility for setting up the project. Because of different schedules, sickness and lack of motivation the level of commitment and ambition varied a lot both over time and

between different members of the group (f.e look at the contributions of Oscar Beronius and Joakim Sehman in week 7, which were clearly superior to the rest) as well as for each individual.

## External tools

### *Firebase*

Working with FireBase was a great decision on our side, we discussed back and forth whether we should set-up our own combination of Node, Socket and some DB but why reinvent the wheel? To quote an answer we found while researching this:

*"I suspect with a node host, socket.io, and some nosql style db like mongo, you could get pretty close... but is there a good reason to? I suspect the end goal is to build something on top of it - why not give yourself the head start?"*

However we also felt that we might be missing out on an opportunity to learn a lot about setting up our own server and interacting with a database from a running real-time application. We think we would have learned a lot from this but since we took too long researching what to use we choose what we thought would be a little bit easier to have the time to create a more stable application with less bugs and without having to worry about stability of the server and database.

Since Firebase as a service is entirely cloud based, it proved to be tricky to write automated tests for most of our functionality. However, the functionality is tested when we use the application and you can view the results in the web client hosting the database. This tests both the application code and the firebase focused code at the same time. This compared to a more "regular" database service has both pros and cons: the pros is that you can view the changes directly in the webclient, eliminating the need for unit testing. This on the other hand is also not entirely a good things since you have to use a web client alongside the regular development tools to view the changes. If you would have used a "regular" database, structuring and adding things with your own authentication it enables unit-testing easier since you can always read from the selected part of the database where you just added data.

### *Slack*

Great communication tool we probably should have spent more time setting up properly in terms of channel structure and maybe discuss how to use it, great tool which we didn't use to it's full potential.

### *Trello*

Also a great tool especially to structure up sprints, we started using it early on with the integration to slack which worked very well. We could have spent more time adding cards and structuring our work boards here as well. We used Trello to it's full extent from week 6 or so after great structuring from Oscar Beronius. Before this we had some issues with people not updating the board properly and we ran into some unnecessary issues due to this.

### *STAN*

We decided to use it at the end of the project and we think it’s a really good tool and will probably use this tool more in the future.

## Reflections and conclusions on our project

We have had to work hard to take this project to a point where we felt satisfied with the outcome, but in the end we managed to achieve our goals better than expected. However, we feel that there were several factors that slowed down our progress and performance in general. If we knew about these things before the project we would have been able to achieve much more with a lot less effort, which has been a very valuable lesson to us.

### *Group dynamics*

To begin with, we never discussed our ambition levels within the group, which led to some group members having expectations that other members did not live up to. This did not have a major impact on the overall performance of the group, but it did cause some stress and demotivation. It also had a certain impact on the group communication which led to impairing the potential effectiveness all group members could have. This was mostly prominent in the middle stage of the project, but changed towards the later stages of the project when the motivation of the whole group increased and evened out, which enabled us to work very efficiently and cooperatively in the end.

The main problem with our project was that we did not have a discussion about roles before we started. We did not think this would be important but we were proven very wrong. To begin with we never appointed a project leader, which led to us being very disorganized as a group. No one of us had any interest in doing administrative work such as structuring up the work progress or booking rooms and setting up meetings on a consistent basis. Since no one was appointed any particular areas of work, except areas of coding, we had problems getting things that were concerned as 'less fun' done. This included keeping up meeting protocols and using well structured work schemes such as scrum.

We think that this was the root of the problems the group had with communication and dividing up work, which could have easily been avoided.

### *Planning and executing*

This project has been a very good experience for us in terms of developing a product from scratch and delivering a finished prototype in a limited time scope. In the beginning, we were very lost in how to get started and what research we should conduct and in what proportions. Since we were not used to this kind of work progress we had to learn to adapt and constantly develop our methods in how we approached problems.

## *Refactoring code*

The single most time consuming part of this project has been refactoring our action bar. In total we have performed 3 major refactors, two of which were implemented during the mid phase of development. The first two refactors were made when we had done some basic coding and developed a greater understanding of how the program should work as a whole. They were both finished in one evening without any problems, and had us very confident in our refactoring skills.

When we implemented the action bar at the top of all activities we realized that we would have to refactor some of the code to make the structure good, but we were naive and thought that it would be finished as smoothly and quickly as the last two refactors we had made. This was a grave mistake, because when the time of refactoring came we would experience that changing the action bar into a tool bar required some fundamental changes in the whole project structure. This proved to be extremely time consuming because we had filled out the project with substantial amounts of code already, and changing some fundamental properties required more changes further on and in turn more bugs that we had a hard time to understand.

We got to a point where the refactor was almost complete but the app started acting strange and unresponsively, at which point we realised that we had to drop the changes and revert back to the old implementation of our action bar. In the end, we did manage to change our implementation to follow a good structure which was very rewarding. This was a very bitter lesson, because we spent an unreasonable amount of time on something that was discarded and that we could have used for better things.

In hindsight, we have learned that if we need to refactor our future projects we should do so as soon as possible and make sure to research the parts involved to avoid underestimating the amount of work such a change would need.

## *Getting started*

In the beginning of the project we spent a lot of time researching how to structure up our code. It got to a point where we kept researching but did not get much wiser on the subject. Doing this, we wasted time that could have been used for better things. One realization this gave us is that we should have spent a lot less time researching and planning things we were unwise of and instead just start coding in test projects and actually learning the area. Basically, we learned that we would get further if we strived to always be effective and doing something, and when we were stuck do something else in the meantime.

## *Conclusion*

If we could start the project over we would probably put way more hours in during the starting phase of the project making sure all tools that are being used are set-up to being a great aid in assisting the development once it begins. In our case setting up slack, trello and making a proper project structure way sooner would have given us the advantage of

focusing fully on code once we had developed an idea everyone thought was good. We ended up spending lots of time researching things without actually testing whether applying them to our project was a good idea or not. This gave us lots of knowledge but all didn't actually help during the development of the project in the end. We definitely should have decided on a project leader or similar who set aside a little development time to set-up necessities such as Trello, Slack and making sure the project group worked more together as a group.

## Electricity Challenge

The workshops in the competition was something our team didn't discuss very much. But what happened in the end was that the member who took initiative first to sign up for a workshop could pick whichever he wanted and then the others could do the same until everyone signed up for attending at least one and all workshops had at least one of the team members signed up so that the group was to attend all workshops. We felt that several workshops wasn't very important for our project and therefore we had a lack of interest in the workshops. The workshops we felt was important we signed up for first and the one who got to go was the one who signed up first as mentioned before.
The rest of the challenge was great, the workshops we wanted to attend was really educational and productive and the communication with the stakeholders were really good. It felt like they really wanted to see some nice results and were very helpful and supportive when we asked about something.