# Reflection report

## Le Scrum Masters

*Melina Andersson, Albin Bååw, David Illiefski, Simon Mare, Pedram Shirmohammad*

# Table of contents

**Application of scrum** **3**
Social contract 3
Roles, teamwork and social contract 3
Time distribution 4
Effort, Velocity and task breakdown 5
Used practices 5
**Documentation of sprint retrospectives** **6**
Sprint 1 6
What went well? 6
What went wrong? 6
How to improve: 6
Sprint 2 7
What went well? 7
What went wrong? 7
How to improve: 7
Sprint 3 7
What went well? 7
What went wrong? 7
How to improve? 8
Sprint 4 8
What went well? 8
What went wrong? 8
How to improve? 8
**Reflection on the sprint retrospectives** **9**
**Reflection on the sprint reviews** **10**
**Best practices for using new tools and technologies** **10**
About git 10
About the scrum board 11
**Reflection on the relationship between prototype, process and stakeholder value** **11**
**Relation of your process to literature and guest lectures** **12**
**Evaluation of D1A and D2** **13**
D1A 13
D2 14

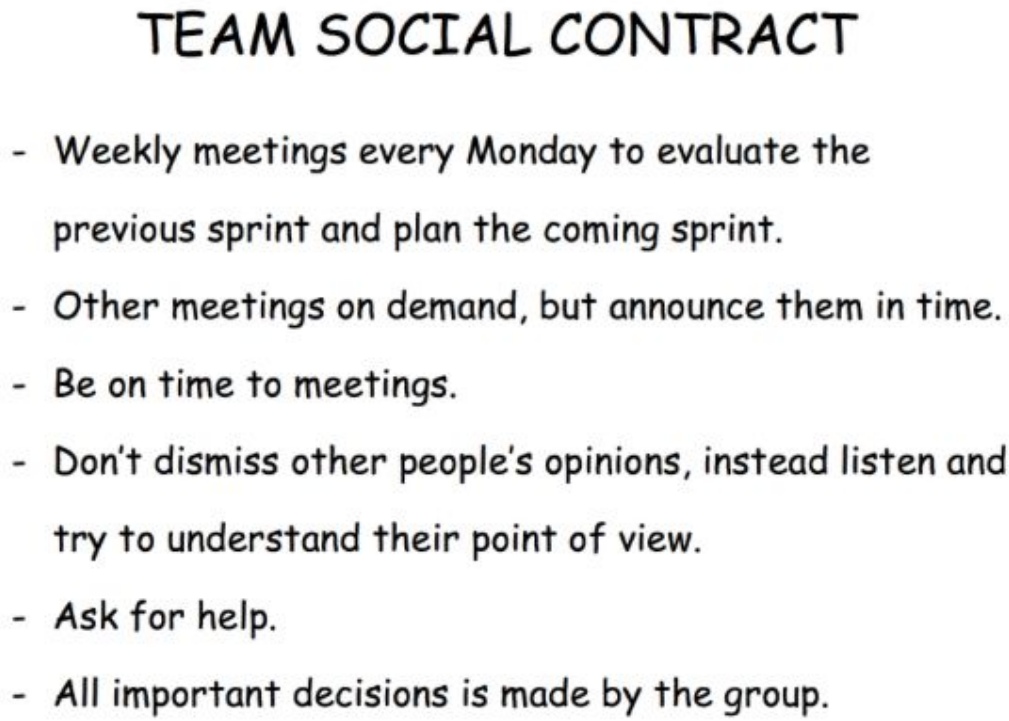
**Burn-down chart** **15**

**Design Rationale** **16**

**Conclusion** **16**

# Application of scrum

## Social contract




## Roles, teamwork and social contract

The ideal way to work was by always knowing about what every team member was supposed to work on. At the start of the project we tried to work with this in mind. After a short period of time we realized that we weren't able to understand what to focus on, nor how to handle new and unexpected tasks that would show up during the development of the application. This was because a significant part of the group wasn't used to develop android applications and using Google API:s.

Not being used to the development tools resulted in a lot of trial and error trying to figure out how to use the new tools and looking for good practices to apply to the project.

Whenever new tasks or questions arose, the team members often tried to deal with them themselves, meanwhile another team member had already solved the exact same problem. This lead to a significantly decreased sprint velocity. The team decided to tackle this problem by defining smaller and more precise user stories in future sprints, to make it more obvious to the developers what to and not to implement. This also resulted in that communication between team members improved and the team members started to mention when they had started working with or completed an assignment.

The weekly monday meetings didn't work out as expected. Sometimes the team missed them due to a lack of communication. To prevent that team missed meetings, the social contract should also have included a specific time for the meetings. The social contract was followed in all other respects, except that a team member could be late now and then.

## Time distribution

In connection to what was written above, the team realised early on that the lack of knowledge in android development lead to a huge amount of questions and unexpected assignments that needed to resolved. Which resulted in a lower velocity than expected in the two starting sprints. Later on the team decided to set lower sprint goals than what it deemed to be suitable, since it knew that unexpected issues would arise. This resulted in better approximations of what the team could deliver in the current sprint.

As the team had a hard time with approximating the time, the team was far behind in the time schedule. Which resulted in that the team couldn't demonstrate their design to the other teams, as the design hadn't been implemented yet. The team tackled this issue by using the wireframes they had designed since before, which resulted in that they could still get feedback on their design. But it also resulted in that the team forgot to work according to scrum. What could have been done to avoid this will be discussed later on.

## Effort, Velocity and task breakdown

Since the application is quite simple with only two views, one showing the map and the other displaying the current point of interest, it was not obvious of how to split the implementation assignments among the team members. It was tough to define the user stories so that they wouldn't rely on each other.

Since our app is very simple, with only a map view and a view for the so called point of interest, it was challenging to split the implementation tasks between five persons. It was hard to make the user stories not to be depending of each other.

The team's work could have been more aligned to the goals of "carpaccio". When team realised this it was almost too late. The team tried to write the user stories according to the "INVEST"-criteria, but the team failed already at the first criteria, independency. To accomplish nine out of ten of the user stories, the two main user stories had to be implemented in a proper way. On top of this the API:s the team were supposed to use, started acting up which resulted in more work than expected. This was all solved by shortening the sprints in order to be able to deliver customer value continuously.

What could have been done to confirm to the goals of "carpaccio", would have been to define what an assignment is and think things through before writing the user stories. That could've saved the team a lot of confusion revolving the idea of the application and streamlined the work so that all the team members could've been programming simultaneously.

## Used practices

When the team first sat down to discuss how to manage the project, the team agreed on splitting tasks and only meet for weekly meetings, with the exception of if a meeting was needed to keep on going. This was complemented by using a group chat for "stand up meetings", where the team would report on day to day information. This seemed like a good solution as every team member has their own schedule, but it may have resulted in a lower

velocity as it's easier to help other team members resolve problems face-to-face. A solution to this could have been practicing "pair programming" or some kind of programming as a group.

# Documentation of sprint retrospectives

## Sprint 1

### What went well?

It was a short sprint. The team delivered a map view as expected.

### What went wrong?

We wrote a user story telling them to implement a system for the assignments, even though we didn't have a clear idea of what the assignment-system was supposed to like. Some changes to the initial idea of the map implementation were made. The major one being a switch from Open Maps to Google Maps simply due to availability and ease of access. Some hiccups with Google Maps API occurred during the initial implementation, nothing major but this still added to the required effort and as such, we surpassed our estimations.

### How to improve:

Estimate effort more realistically next time. Aim lower, hit higher. We also need to articulate our user stories more clearly. We should put effort towards the design and implementation of our assignments and how to express them without the use of text.

# Sprint 2

## What went well?

Not much. Due to bad planning time wise few tasks were accomplished during this sprint.

## What went wrong?

The sprint was planned during a weekend which in hindsight was a bit overambitious. This lead to a lower velocity than expected. The SQLite database which was supposed to store the apps data was written but not fully integrated with the android studio project files and caused crashes. These were never resolved and different approaches for storage were implemented.

## How to improve:

Sprints on weekdays from now on, we have jobs and other activities planned during our free time, this needs to be taken into account. Finish the integration of the database or alternatively, look at other solutions for data storage.

# Sprint 3

## What went well?

The app actually delivers some value as of this sprint. Things are happening when opening the map.

## What went wrong?

Things are happening, these things are not always the right things. While the app starts to look a bit more extensive than an example project there is still a lot missing. The things added are not really tied together as one would wish for at the end of a sprint and the deliverable of

this sprint still would have needed a lot of explaining comments if shown to a product owner. “This and that exists in the backend but you can’t see it yet”- kind of stuff.

### How to improve?

Focus on delivery value. At this point we were working a bit independently of each other and not really tying our additions together in a logical manner. This is an obvious and straightforward area to improve upon.

## Sprint 4

### What went well?

- For the first time we have a useful product. The product is by no means finished, but the most fundamental features can now be viewed inside of the application.
- The sprint was longer than the other sprints. Having a longer sprint made it easier to achieve a working product in the end of the sprint, which is the goal of every sprint.

### What went wrong?

- There were a lot of things which had to be done during this sprint to achieve a useful product. Which meant that there was a huge gap in the time needed to achieve the goals of this sprint compared to the previous sprints.

### How to improve?

- We need to get better at planning our sprints, to make sure that all the sprints need equal effort. This can be achieved by having longer sprints with more user stories to implement, or splitting up long sprints into multiple short ones with less user stories to implement.

# Reflection on the sprint retrospectives

We chose to work with google maps although we knew that it would become a problem later on since Google has some restrictions for which areas of their API's that are publically available to use. There are other great open source map-API's that does not come with these type of boundaries, but we thought about the value proposition - Google maps is much easier to implement and we would deliver something of value much faster. The main functionality we would be lacking due to this choice was the offline mode, Google does not want outdated maps with their logo on them so they reserve offline caching to themselves. We realise that it is a very important part of the app for the main persona, a newly arrived with limited or no 3g at hand. However, we compromised and reasoned that with the restricted time we had to develop. This was to be seen as some sort of "beta" version of an app that later on could be further developed to achieve the maximum potential. In regards to usability, the app would function exactly the same whether in online or offline mode, so for prototyping purposes this was an easy choice to make in order to speed up the process.

In the first sprint we had the user story about implementing the assignments. Our first obstacle was how you express an assignment in ways other than text. This made us rethink the entire assignment part of the app. Together with Sebastian, our design student, we figured out a new solution (which of course led to a different velocity than what we had originally estimated for this sprint). Therefore this user story was never done in the first sprint.

After some hours spent on a troublesome database we set out to find another suitable solution for storage. This, in combination with a decreased amount of data to be stored, due to the mentioned changes on how assignments would work, let us use a simpler way to store our data.

# Reflection on the sprint reviews

The only sprint reviews we had, happend every wednesday when we displayed to our IxD-student Sebastian what we had accomplished up to that point. Sebastian gave us feedback on the user experience and functionality of the application. Sebastian also gave us feedback on the application which he had received from a girl who moved to Sweden a couple of years ago.

For every week that passed, our vision of the application became clearer with the help of Sebastian. As he would have us rethink our vision every time we met. We are thankful for the cooperation we've had with Sebastian and the other IxD-students.

# Best practices for using new tools and technologies

## About git

In the start of the project we tried to all work on separate branches, as we had different coding assignments. It didn't take long until we realised that the features we developed built on each other. This lead to some problems and bugs when merging these branches into a collective one over a short period of time. From that point on, we developed everything in one branch separated from the master branch, and merged with the master branch only when we had a stable build.

A better way to use git would be by using a "develop"-branch by the side of the master branch. You would then have "feature"-branches on top of the "develop"-branch to be able to push without a tremendous amount of conflicts but meanwhile be able to build on top of features implemented by others.

## About the scrum board

We used a tool called Trello(https://trello.com) to setup a virtual scrum board in the start of the project. In the begin we had very broad and hard to define user stories. This lead to that the user stories needed a high amount of time to implement and that the group members often found themselves wondering about what they were supposed to implement. In retrospective the user stories could have been split up into to smaller ones, which would have made them easier to implement and more clear to the team members of what to implement.

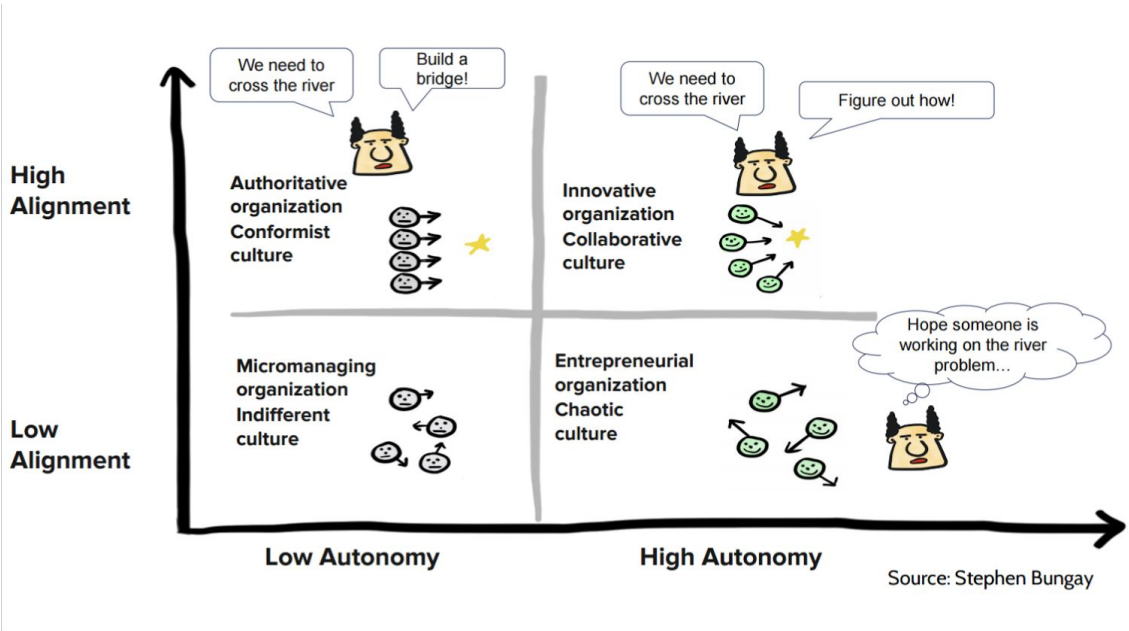
As the user stories were too broad, the scrum board was rarely used as one user story could take weeks to implement. With smaller user stories the scrum board would have been used more often, as the developers would have to go back and mark user stories as completed and elect a new one.

# Reflection on the relationship between *prototype*, *process* and *stakeholder value*

During the project, we tried to deliver small chunks of stakeholder value continuously. What we could have thought of is aiming for a prototype of each element in our app to start with - instead of thinking that we could make everything complete at the first shot. It was quite irrational thinking that we were going to deliver a perfect map view after the first sprint, or a fully implemented “point of interest” view when we did that user story. So when Håkan told us that the finnish students had this “prototype goal” at the lego exercise, we realised that it was what we actually did (delivering a prototype of each element for a start) but we didn’t write the user stories according to it. When a prototype was made, we acted like “now we have a map view, that user story is done” but the map view required a lot of additional work, which we afterwards didn’t estimate any effort to. So for the next sprint, we put some new work on the table, trying to deal with the invisible work we had hidden simultaneously.

# Relation of your process to literature and guest lectures

We drew a parallel to Maria's lecture in D2, about not knowing exactly what you want so you do as little as possible, doing it in a way so that you can get feedback on it and then implement things more when you are more certain of how you want it to look.
If we had gotten further in our development we could have used us of the testing methods Michael talked about, for instance, A/B testing. Our only testing session for getting user feedback was at the demonstration on Oct, 12th. We were planning on going to a "Språkcafé" in town to ask foreigners what they thought about our app,  but when the day came about a week before the final presentation, the design was still so much behind that we realised that it would not give us anything. In order to give feedback the user must at least be able to see what the app is all about.




We somehow can relate to this slide that Michael showed, sometimes the lack of communication made us think like the person in the bottom right corner. Maybe some kind of team leader delegating tasks or a strict schedule for daily scrum meetings would have helped

us knowing who is doing what and when that must be done, so that someone else can start doing that thing that builds upon what the first person did.

Jens talked about digitalisation which of course has relations to our project in a larger scale. Software is no longer only a part of the society, it is integrated in everything around us. That includes teaching and learning which is why an app like this could be a gateway to learning swedish and integration.

# Evaluation of D1A and D2

## D1A

The lego exercise gave us insight to scrum and what the typical beginner's mistakes were. The ideals we brought with us into the project were:

- Start with the most prioritized task
- Organize the work well so that no task is missed (or done twice)
- Have continuous contact with the product owner, stakeholder or anyone else who has interest in your product.
- Focus on quality not quantity, and don’t underestimate the amount of time that takes.

All in all we followed these points rather well but in the end we may have forgotten about the last one, when we realised how much we had left to do. The reasons for this will be covered more in the D2-reflection. We also didn’t put as much effort into having a continual contact with the eventual users of our product. As it always seems to be, starting out, we had good intentions regarding this. We were going to go visit an accommodation for newly arrived but that visit got canceled due to things outside of our reach. Instead, we were going to go visit a “språkcafe” but this planned was scrapped when we deemed our product to be lacking of content and we didn’t feel like showing up with only wireframes to get feedback on.

When looking back at our original thoughts written in d1a, we feel like they were valid from the start. If we only had stood by them more closely, our project might have turned out more polished.

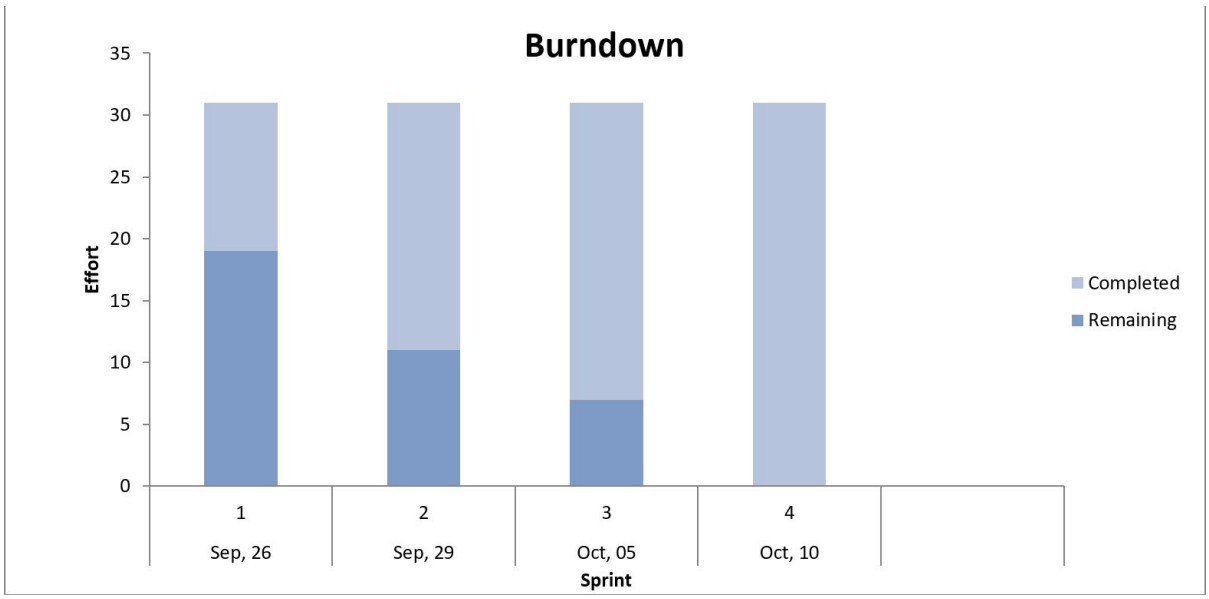
## D2

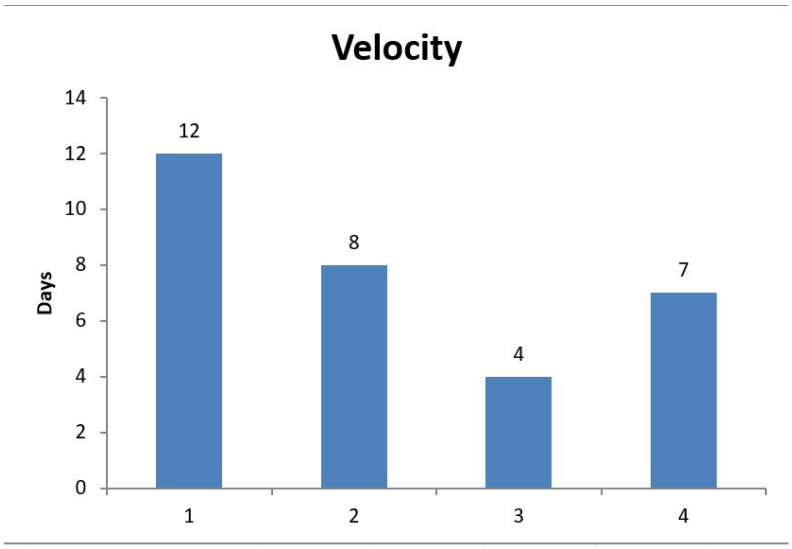
In our half time evaluation we talked a bit about how we had reached a somewhat final idea of what the project was supposed to become. How we had rocked back and forth between different versions of similar ideas and how we, following this, hadn't begun to actually build a lot by that point. In hindsight, not building the wrong thing was in some respects the right choice. This was the same conclusion we came to in the half-time evaluation. However, this also led to us starting the actual building process somewhat late into the process which at times, made it all feel a bit stressful when coding towards the deadline/demo day. This however was somewhat unavoidable since we had just "finalized" our vision of the project by the time of writing D2.

Also mentioned in this evaluation was our plan to start working with shorter sprints when building. This to give ourselves continual, lesser deadlines in order to make the most of the time we had left. This was for the most part a good foresight on our part. While our initial reasoning was that we would let us review our progress more often, the largest benefit of this was instead that now we had the chance to review our mistakes more often. As written in the sprint reflection, these sprints weren't always perfect. However, when you stop, look at what you just did, evaluate if it was beneficial or detrimental to the project, you can more easily improve your coming efforts.

Thus, the conclusions we reached in our half-time evaluation still stands, although with some slight modifications and with the addition of some of the obvious drawbacks of starting building late into a project.

# Burn-down chart




As mentioned earlier, after October 10th it got so stressful that we just worked on the user stories already marked as “complete” which needed some rework, mostly on the design, and some new things that came across after time. We didn’t put time in formulating new user stories with estimated effort for these problems, and that is why the burndown chart looks like we were finished at October 10th.

The velocity chart is not that accurate either since, as we already said, our first sprint backlog contained the two main views and not just prototypes for them. If we had written and estimated them differently and if we documented a fifth sprint our velocity chart would probably have looked exactly the opposite.

# Design Rationale

Our app relied heavily on Google Maps- and Google Places API. We choose Google Maps mainly because it's well merged with android and it would simplify the coding for us. Both Google Maps and Google Places also have very good and simple methods, like marker customization, sorting places and receiving right type of places.

The communication between classes isn't perfect. There are some methods that are public static which sometimes unnecessarily exposes them to other activities/classes. This could be better sorted with a Singleton pattern. We initially tried to setup a local database using MySqlite but later due to lack of time chose to save data (such as completed places) in settings and to keep all video and images raw in the application. The raw material made the application very heavy and it would of course have been more logical to keep them elsewhere and fetch them only when needed.

# Conclusion

The team had a blast and learnt a tremendous amount while working on the project. We got to see a "real" project through, and learned how effective teamwork is not always easy to achieve. Most importantly the team learnt how to not work in the future.

All-in-all the team is celebrating the success of the application and the project revolving it.