Reflection Report
Scrum Project

Software Engineering Project

Android Application - Meet a Sweedt

Group: Untitled

October 27, 2016

Contributors:

Name	CID	ID number
Ajla Cano	ajlac	941104
Arvid Hast	hasta	961201
Daniel Hesslow	danhes	951113
Niklas Jonsson	nikjonss	961010
Fredrik Lindevall	frelinde	960222
Eric Shao	ericsh	960319

Contents

1 Introduction

We were given the task of creating an Android application that would assist newly arrived immigrants in Sweden with learning the native language in an effective way. What started off as a simple and humorous rhyme off the top of our heads soon became a reality as we decided to develop the idea further.

Throughout the project, members had to learn how to use new tools and apply them to our application as well as work together in a new environment. Not only did we learn software–such as Android Studio–from scratch, we also had to master the basics of running a database together with our code.
This project focused mainly on the concept of Scrum and how to work around it. Compared to older projects that we have participated in, this project felt vastly different. All six members of the group were unfamiliar with this new concept–as evident when we first started off–but along the way we adapted and learned how to fully use Scrum to our best advantage. Each member of the project group had their own unique strengths that contributed towards completing the project in an orderly yet effective way.

"Reflection is an assessment of what is in relation to what might or should be and includes feedback designed to reduce the gap."

- *R.Smith*

Not everything went as planned and in this reflection report we address all the things we have learned from. We talk about working with Scrum for the first time, change of plans, stake holder value, vertical workflow, and more.

2 Application of Scrum

Scrum, as a new way to work, was interesting to learn. We experienced difficulties but also realized that there was indeed very good points with working this way.

2.1 Used Practices and Time Distribution

Our application of Scrum has been straightforward. We've had one Scrum meeting every week at the end of a sprint which also was the beginning of a new one. The meeting consisted of review and retrospective of the last sprint and planning of the new one. We reevaluated the backlog and delegated new user stories and things to do to all project members.

We had quite a lot of difficulties during the first few sprints. Since the project was still very small it was hard to find things to do for each member that didn't cause a conflict with someone else's work. The fact that we didn't have any experience working in Scrum also caused some problems and we didn't fully grasp what it meant to work in vertical slices. After the first sprint we had a working application to continue to build on but we also had some work that didn't add any customer value at all yet.

Eventually the Scrum meetings became a lot smoother, both because we learned how to do it right and because the project became bigger and working in vertical slices is a lot easier when you already have a working front- and backend. In retrospective it would help the work a lot throughout the project if we organized the user stories in a better way so that not so much time would have to be spent finding relevant high-priority things to do while maintaining the vertical workflow. We should also have tried to find more time to work together in getting the first vertical slice of the project together instead of trying to complete one part each since we ended up having all parts but with no time left to integrate them into one unit.

At the beginning of the project we decided to have one Scrum meeting a week but now at the end of the project we realized that some things would have been easier if we had those meeting more frequently, since communication and integration is very important when working vertically. That is absolutely something we have learned and will take to our next future project.

2.2 Roles, Team Work and Social Contract

Between our Scrum meetings we've kept in touch with each other daily both through a group chat and with meetings where we worked together and

helped each other out with problems that came up. A part of the work was done by members individually, since all members had different schedules. If any bigger problems and difficulties appeared, then the members met to solve it in pairs or the whole group. The meetings and the chat was more used as a tool to make sure that everyone had something relative and high-priority to do and that the work continued smoothly.

During the scrum-meetings we asked everyone how it was going and as soon as a member needed help with something we immediatelly found another member that could help with that particular problem efter the meeting. So the whole process was that the members worked both individually but also in pair/smaller groups. If the group ever faced bigger problems and questions then the majority met up to solve it together so no one misses any potential changes.

Our social contract ended up working well. Since most members knew each other before working together we could assume that everyone would take responsibility for their own work so the social contract doesn't have many specific strict demands. If we didn't know each other before or if the group consisted of more members it's likely that we would have to be a lot more specific to avoid conflicts and make sure that the work progressed smoothly.

2.3 Effort, Velocity and Task Breakdown

After alot of discussions we decided on a velocity of 18 velocity points (VPs) per sprint, or 3/person/sprint in our group of 6 people. What we learned from the Lego-exercise was that we had to have in mind that everything won't go perfectly fine so when deciding the velocity we were less optimistic than during the exercise.

We used Trello as our Scrum board and broke down the project into user stories and then the user stories into smaller parts (tasks inside the Trello cards). This was something that worked very well in the beginning but unfortunately as the time went and more and more stuff came up to do, we had a tendency to forget to break down everything into smaller parts and just jumped into the user-stories without thinking it through. This did cause smaller problems in the communication and the idea of what exactly was being worked on. If we had a chance to re-do this, we would definitely try to break down every task from the beginning so that absolutely everyone was on the same page.

3 Sprint Retrospectives

During the first sprint retrospective we had a lot of things to improve. After the first sprint we had a better idea of how we could break down the work into user stories and the Scrum board was improved a lot. Each member of the team knew what needed to be done to create the project's base, as well as a rough time estimate and effort needed to complete each task. The sprints following that helped give us a tool to reflect back on into our project. Although significantly shorter time was spent on consecutive sprints after the intial one, we knew that these meetings provided the team a solid insight as to where and how well we had worked with the project. As time went on, it was clear that a couple things needed to be changed to accomodate the changing pace of our work process.

One thing that became apparent later in the project was that the biggest factor to why the number of VPs we could complete each sprint fluctuated was not bad estimations of the work required to complete a user story but rather that our velocity changed. Due to several factors; different schedules, other courses having a fluctuating workload and maybe most significantly, the motivation increasing the closer we got to the deadlines, our velocity during the last couple of weeks was significantly higher than during the first.

Our ability to estimate the work required to complete different user stories improved more and more when we learned from previous work and could compare new user stories to previously completed ones. To actually figure out a reasonable consistent velocity we would have to be way stricter when it comes to the schedule and how much work we put in each week. The end result of the project is close to what we envisioned and we think that the work required to get there was reasonable, but none of this is actually based on collected data. It is entirely possible that we put in less work that we think and that the end result could be better, or that we misjudged the work required to reach the desired result and worked harder than agreed. By measuring the time we put into the project each week we could avoid guessing and actually know exactly how much effort it took to complete something.

4 Tools and Technologies

The main tools and technologies used in this project were:

- **Github** - a Git repository hosting service we used for managing and storing revisions of our project. This tool was mainly used for the coding part of our project. Since everyone in the group was already familiar with Github, we never experienced any difficulties with using the tool itself. There were some merging problems which is a very common thing to experience, so with a little patience and teamwork we solved those merging problems. Github was used on a daily basis.

- **Google Drive** - a shared folder online. When it came to the documentation of the project we decided early on to do that in our shared Google drive folder, where all our deliverables, pictures and notes were stored. Goodle drive was used once a week.

- **Trello** - a collaboration tool for organizing projects. This tool was a new experience for all the members. We started by dividing the board into different cards such as *Project backlog (User stories)*, *Sprint backlog (To do)*, *In progress*, *Testing*, *Done*. One thing we discovered and thought was very good, is that we could assign concrete stories to each member so that everyone could see who was resposible for what in case of questions. Another thing we used was color coding. The thought of this was to see the ratio between all the different parts that cover the vertical workflow. This helped us in the beginning when we planned the user stories but also on our weekly Scrum-meetings when new user stories needed to be assigned and we had to think of the vertical workflow. We tried to always work on all the parts but it worked well some weeks and less good other weeks. Since our group consisted of a mix of people with different skills and stregths, Trello helped everyone pick and choose what they deemed was suitable to their own set of skills. It helped highlight the strengths of each person. Trello was used once a week.

- **Android studio** - a tool for code editing, debugging and app building. Everyone in the team, but one member, were familiar with this tool and knew what to do from the start. While some members prepared the project in Android studio, some started looking at databases and others simply began getting familiar with the tool. After a short period everything was set and everyone was ready to start on the user stories. We tried to implement as many help-classes as possible which faciliated the implementation of our application since we knew exactly how everything worked. Android studio was used on a daily basis.

- **Server And Database** - The application is built using a three-tier architecture. Every client handles presenting the information to the user. It communicates with a server who processes the requests and and in turn communicates with a database to aquire the neccessary data. The server is just a java program running on an old laptop. The database is called mariadb which is binary compatible with the more famous mysql. The reason of using mariadb instead of mysql simply comes down to installation process, which on arch-linux was way easier for mariadb.

  A three tier architecture is neccessary if you wan't to seccurely handle passwords and other sensitive information. Skipping the server part and directely interacting with the database from the client would have been easier but it would also mean the the database needs to be open for everyone to read which is very bad if you have information that needs to be secure. While we don't claim that the current application is completely secure it is set up in a way so that it could easily be corrected for in the future.

5 Prototype - Process - Stakeholder value

Throughout the project, we received and applied plenty of input from another party involved in interaction design. At first, our prototype that we created from a simple brainstorming session was lacking many essential characteristics that the stakeholder value approach contained. For starters, we needed an incentive for people to even use the application to begin with. Secondly, we had to keep in mind what our customers needed and what our goal was as programmers. Lastly, we needed to take the input as well as feedback from the design team and apply it in a flexible way to our project.

Ever since the beginning of the project we have had a vision of how our application would look like and how it would help our problem in question, that is: help the newly arrived to learn the swedish language.

During the project, our vision of the application and our user story priorities changed a lot. A lot of this was because of discussion with the design team after they were in contact with potential users. We learned that there were a lot of potential newly arrived users, but that the real challenge would be to get swedes to use the application as well. Most swedes they talked to said that they would be willing to use the application, but only if doing so was fast and easy and if there was something in it for them as well. There are plenty of newly arrived people who wants to learn Swedish and become a part of Swedish society and culture, but we can't expect the same amount of swedes to be willing to spontaneously take that much time out of their lives without getting something out of it themselves.

This realization made us rethink several parts of the application. In the end our application might be working perfectly and have a lot of cool features but without users, both learners and people who already speak Swedish, no customer value has been added at all. Firstly, we made it a top priority to make the application fast and easy to use. This included the ability to quickly match with someone and engage conversation through the chat. We also increased the priority on message translation and the ability to find fika-places within the application so that everything that is needed for two people to start talking or meet is quick and accessible without having to use other services or applications and decreased the priority of additional features like the ability to create and invite friends to events.

Finally, we rethought which two kinds people would establish contact. In our initial vision the idea was that someone born in Sweden would talk to someone newly arrived. We decided to change this to make it possible for any two persons so match, but sort other users in the matching view based on a couple of factors. These factors include among other things the distance

to the other user, interests they share and most notably their knowledge of the Swedish language. Now any two people could find each other with the application, but someone who is not very good at Swedish yet will have a higher chance to find someone that they can learn from and vice versa.

The goal is to have an equal amount of Swedish speakers and learners and there are other additional steps that could be taken to increase the incentive for Swedish speaking users to use the application if we would continue to work on the project. The design team suggested some kind of progression for Swedish speakers as incentive to use the application and we also had the idea to talk to cafés and see if they would be willing to offer fika at a discounted price if customers visited the café as a result of using our application.

6 Literature and Guest Lectures

How the literature helped us in the project
Literature wise we used the links on the course home page, mainly to read more about Scrum and the concrete parts we really need to think about. On Gojko Adzic's article (Breaking a feature into tasks @ github.com/hburden) about the **"hamburger"** we tried to follow the six steps he is explaining.

- Step 1 was to identify tasks. Adzic describes that as technical steps on high level which we interpreted in our own way and wrote down on a paper during our first scrum-meeting (or what we thought was a scrum-meeting but turned into a regular planning-meeting). Instead of drawing a hamburger, we did a mind-map where we wrote "Profile", "Chat", "Matches", "Map", "Event planner" etc.
- Step 2 was to identify options for tasks. Instead of splitting us into smaller groups to discuss qualities for each task, we ended up doing it together where we processed every area individually and wrote down the smaller headlines and how to implement everything in the best way possible.
- Step 3 was to combine the results from the smaller groups. Since we did everything together we couldn't really do this step as he describes it so a downside with that was that we only had one perspective of what the "best implementations" were.
- Step 4 was to trim the hamburger which basically means to compare lower and higher qualities and rank them after how difficult they would be to implement. We only partly fulfilled this step by ranking the qualities after how important they were to implement so our customer could see some kind of product early on. What we learned from this was that we should have done the comparison of difficulty to really grasp the workload from the beginning.
- Step 5 was about taking the first bite. Since we didn't talk about the difficulty for everything before, we did that when we delegated the tasks to each member and tried to cover as many task as needed to follow the vertical work flow. When doing so we tried to hold the same level for the majority of tasks and if one task was important and more difficult than others than that particular member had the right to work with said task during several sprints.
- Step 6 is just to take another bite and then repeat, which was something we tried to follow.

How the guest lectures helped us in the project
All the different guest lectures talked about different aspects. Working agile is something that majority of the industry does and that gave us understanding in how important it is to actually try to learn to work so in our project.

Another thing we learned from the guest lectures is that things change and we always need to be prepared for that. We need to work in a way so we can adapt to those changes without hurting our project too much. The best way to do so is to have this in mind while planning a project but also developing all features inside of it. In real life technologies and trends can change all the time and it is important to understand and predict that.

"When things change due to disruption, we need to predict and understand the massive change to come. This is to understand the new uses cases and behaviours to be able to design solutions for future needs"

- Joel Rozada, The Techno Creatives

That quote was something we really tried to work after everytime we implemented things for our application. We left room for changes in the application itself but also when planning we tried not to be too optimistic in case of unexpected events. One example of that is when we implemented our help classes to support the different features in the application, we tried to write everything as generic as possible while leaving room for new things to add if needed.

7 Evaluation of D1A and D2

What we learned from the Lego exercise and D1 was really helpful throughout the project and prevented us from making a couple of mistakes. We valued the customer input we got from the design team and their work and as a result changed several things about what we focused on in our own work. We also kept in mid to focus on the important things that the customer would value instead of getting carried away with excess work.

We also tried to be flexible in our estimations of the workload, even though we failed to factor in how the effort we put into the project changed from week to week. In retrospective we should have been stricter with how much work we did each sprint. That way we would have had a better view of the project progression as a whole earlier and been able adjust priorities based on that instead of just increasing the pace later in the project to make sure the features we wanted to include were ready.

What we had learned after the half time review didn't change the product as much as it changed the way we worked. After property constructing user stories and making sure everyone was on the same page and aware of what was needed both form them and the group as a whole, our sprint meetings and the sprint itself went really smooth.

One mistake we still made regarding the product halfway into the project was to underestimate how much work integration of features would take in comparison to the actual development of the features. The vertical slices of our application are quite deep since most features includes everything between a user interface and communication with the server and the database. To make sure that the logic we added to the application both communicated with the server in an efficient way and also behaved as expected in the user interface definitely sometimes added unexpected work and problems we didn't consider when we created the user story.