Software Engineering Project

Final individual reflection
Christer Sonesson

- **What do I want to learn or understand better?**

(A) During this course I have gained plenty of experience and learned a lot of new things, both practical and theoretical. The practical aspects were learning about how to develop programs for Android systems (the most difficult part of which was to set up the emulator), gaining more experience in using github repositories and using Trello for the first time to set up the user-stories, tasks, backlog, and such.

The theoretical aspects were learning about the Scrum-method and how to apply it (and gaining experience in working by that method) and further gaining experience in working with software projects as a project team.

(B) Android development is interesting and is something that I'd like to do more of in the future. I have a gap in my knowledge regarding Python that I would like to fill as well, gaining proficiency in different programming languages is a goal that always seems worthwhile to me.

Regarding the future of the course I would suggest that future projects are based on more "robust and known-to-work" programs, rather than more experimental programs such as the emulator we had to work with in this course. I had some unique experiences with the emulator that I will detail in the next section, a lot of my time in the (quite crucial) early period of the project was spent trying to get the emulator to work, which complicated things and put me behind the rest of the team, and since the program was in such an experimental state it was very difficult to find the correct solution (which would not have been a problem if this was something experienced by all of the team in tandem, but as it was I was randomly singled out and put behind the rest of the team).

(A->B)Both my goals (I won't comment the suggestion further) are personal in nature and easily achievable, I can simply develop Android apps in my spare time, or take some time out of my day to read up on Python (and try to develop small programs using it).

- **How can I help someone else, or the entire team, to learn something new?**

(A) I do not feel that I had any need to help anyone in the team learn anything specific, all information that was needed was accessible online or was learned through experimentation, except for one thing: the problems regarding the emulator on my system. This seemingly unique problem (when asking the entire class online nobody else had had the same problem) helped to teach the team on the requirements and limitations of the emulator that we were given (and what was learned could be applied to similar problems in the future).

I tried a myriad of solutions by myself, including setting up a virtual Ubuntu-distribution in virtual-box, tinkering with a large number or settings in both windows and the virtual Linux-setup, changing the BIOS-settings, etc. After exhausting all possibilities I could think up (including asking the team), I asked online for support from the rest of the class. This gave a number of new suggestions which I could try, most of which didn't work, but finally I stumbled on the correct solution, which was installing it on my girlfriends laptop which was using windows 10 (I was using windows 8.1).

My issues regarding this could help the team, and the rest of the class, to learn about what specific combinations of hardware and software are required to get the emulator up and running (but since I seem to have been the only one that came upon this problem that lesson is not so valuable in itself). The best thing to take away is probably that Android Studio has weird requirements and lacks support for a lot of systems (both hardware and software), so if you are forced to use that program then be prepared for problems and have other systems available.

Since other team members have taken this space to discuss their preferences regarding team meetings, I will add my feelings on the matter here as well. Some people on the team strongly felt that it was valuable to meet and develop in person, and I strongly oppose this viewpoint. I was a heavy proponent of dividing up the work at the start of the week, and then letting each member finish the task as he saw fit, while being able to ask for help online at any time. If people want to meet during the week and do their tasks together, fine, and if people want to have a face-to-face Scrum-meeting at the start of the week, also fine. I am not a face-to-face person if I can help it, I function much better as a screen-to-face person.

What ended up happening was a hybrid system where we kind of divided up tasks at the start of the week, and then had a voluntary meeting mid-week for developing together, and had either face-to-face or online Scrum meetings. The issue is that dividing up tasks, Scrum-meetings and development meetings ended up bleeding together, which put people who did not attend these voluntary mid-week meetings a bit out of the loop, leading to a bit of confusion regarding what to do and fostered a “clique-ish” environment.

(B) I will here focus on how I envision the perfect team-dynamic in these kinds of projects (as in: university projects, not actual work projects).

A meeting at the start of the week, either online or face-to-face, where weekly tasks are divided up between the members. During the week it is up to each member how he finishes the task, and members can freely meet and develop together if they so wish, as long as these meetings are strictly about developing. At the end of a week, a scrum meeting is held, preferably online since they tend to be not overly long and it's a waste of time and money to spend X-amount of time commuting just to hold a 15-30 minute Scrum-meeting. The key here is strict separation between dividing up tasks and scrum-meetings, and the actual development meetings. In this way everybody should be happy.

(A->B)Difficult to say. In student groups there is no clear hierarchy so everything has to be worked out by consensus, there’s often no appointed leader or strong will that can settle things. In work situations I would think this problem is not as severe, since there’s most probably an appointed team leader/manager that the rest can rally behind. The best solution would be to have a clear hierarchy, a leader that has the final word.

- **What is my contribution towards the team's application of Scrum?**

(A) I was a Scrum team member, not a Scrum master, and thus did what was decided on at the start of the week. At the start of the project I helped with filling the Trello-environment with appropriate tasks, and after the period where I was troubleshooting my emulator I did those tasks that were appointed to me at the start of the week. At the end of the project I was a big part in finalizing the look of the Trello-system (backlog, tasks, etc), so that things were correct and could be understood at a glance.

(B) I think I have a good working knowledge about how to apply the Scrum-system, so I guess the final test would be to apply this system in an actual work environment. So that's the final frontier regarding this for me, actually using scrum in a real work-place on a real work project.

(A->B)There is no clear way I can achieve this on my own, so the only thing I can say here is that I have to see where my future takes me. During the course it was said that Scrum is common in Gothenburg, unfortunately I do not plan to work in Gothenburg since I miss my friends and family back home too much to settle down away from them, I will try to work closer to my hometown.

- **What is my contribution towards the team’s deliveries?**

(A) I will here disregard the work I did regarding the Trello-environment and the meetings I attended and just focus on the code-related tasks that I was directly personally responsible for.

1) A system for saving the score to the Android device in a way so that the score can be retrieved later.

2) A system for retrieving the saved score and loading it to the program, so that a user can keep a running tally of his total score from day to day.

   These tasks seems deceptively simple, but when working with new tools and developing for a new system things are not as simple as they might seem. I solved the saving and loading by creating a text-file in the Android storage to which I simply added or loaded the score. The difficult part was wrangling the correct privileges from the Android system to be allowed to actually do this, it was this part that ate up the lion's share of my development time. I don’t think I should go into specifics here, but it involved a lot of googling and a lot of trial and error.

(B) Where do I want to be? The answer is that I want to have done more. My issues regarding the emulator took away much needed development time and I feel that I slipped permanently behind the rest of the group because of it. I have done more than some in the group, but less than others, and I wish that I could have a few more feathers in my hat to show at this point strictly regarding the code (I am satisfied with my effort overall though).

(A->B) The main thing to take away from this is: Don’t have problems with the emulator. Failing that, the other thing to take away is: If you have problems with your software and nobody can help you, try random things even if they seem stupid (like installing your software on crummy old laptops even though your system is more modern in all things except operating system). If it’s stupid but it works, it isn’t stupid.

I also think that more could be done if the team had a clearer hierarchy.