Mini Project SOE

Participants: Christoffer Ilsø Vinther, Rasmus D.C. Dalhoff-Jensen

Part 1.

This semester, our project has several features that would be important to consider when choosing a development method.

- It is a multiproject, meaning several groups will be working on the same project, which will produce a common result. This result will then be used next year by a new set of software students.
- The software must be designed to ease communication with children with Autism.
  - The idea is to develop an Android-based solution, which will provide the children and their caretakers with a number of tools to improve their situation.
- This semester, as opposed to the last few years, we will be having an external customer instead of having to use ourselves as the customer.
  - This customer will not be available on a daily basis, but we will be able to contact her.
- As part of a general meeting in the multiproject group, we have decided that all groups should use the same development method, and that it should be agile.

Choice of development paradigm

The argument for choosing to use agile methods for the multiproject are that we have to continue on a project that was started by other people on a previous semester, and that we will have to work with a subject that we are expected to have no experience with (Autism).
If we were to use a traditional, static development method instead, we would have to do a lot of research together in the whole multiproject group, before we could divide the work into smaller groups, as we would have to establish the architecture through the analysis and design phases.
As we have limited time, we felt it would be better if we made our research on the run, by getting information from our customer. This way, we would also be able to handle changes, if our customer should change their mind on the specifications. As the agile paradigm handles this way of development, we decided that this was the right choice.

Choice of agile method

After deciding what paradigm to use, we have to narrow it down to what method to use. We considered two different agile methods, XP and SCRUM.

Our main concern on using XP, is that everyone in the group has to be equally skilled in programming, writing and such, which our group isn't. Another concern is that XP is not specifically designed for working on a multiproject.

As such, we chose to use SCRUM, as it is more suited for our needs. The method is suitable for changes, and can be used in a large group divided into smaller groups, a so called SCRUM of SCRUMs.

Part 2.

We chose to describe our project using a SWOT analysis.

Strengths:

- All members of the project group have a great understanding of the problem, as our customer agreed on our design, and thought it was very good.
- All of the group members have cooperated in a lot of projects together, giving us an understanding of each others strengths and weaknesses, as well as establishing a team mentality.

Weaknesses:

- The group as a whole is not very skilled at programming, so our progress is often rather slow programming-wise.
- So far, we have had quite a few people call in sick. We hope that this will improve over time.
- We spent too much time in the beginning of the project trying to understand the project from the previous semester.
- We did not contact our customer until our first sprint. Luckily, we did not have to change much of our design, as our idea matched well with our customer's.

Opportunities:

- Since we are working on a multi-project, and we have the source code from the previous semester, we have a chance to improve productivity. If we are working on a part of the project which happens to be easy to canibalize from the previous project, we can implement it and thus save time.

Threats:

- We are depending on the work of most of the other groups, so if they are delayed or decide to change their approach in a way that no longer suits our part of the project, we will have to make changes to our project, which will cause us to loose some productivity.
- We are rather new to testing on the run, so we might have underestimated the ammount of resources that goes into making sure that the system runs smoothly and without major bugs.

Tools and techniques:

In addition to the tools and techniques given by the SCRUM development method (stand-up meetings, having a SCRUM master, sprint backlogs...), we have chosen the following additional methods to improve our productivity:

- Since not all members of the group are equally skilled at the various fields of development,

we have decided to use pair-programming. We feel that we work better this way, and this ensures that at least two people knows how a specific part of the code works, if others have questions about it. This approach has also proven rather useful during the last couple of semesters when it comes to writing the project report.

- As it is important that the students from the next semester will be able to understand our code, we must ensure that it is well written. Therefore we will focus on refactoring our code. Refactoring is a technique in which you transform badly written code into good code. It involves finding pieces of code that are either similar or identical, and replacing them with method calls instead. The reason for doing this is that often enough a specific function needs to be changed, and it is much safer to change something if it only needs to be changed in one place.
- As a multiproject, we need to be able to work together with the other groups. In order to do this, we use assisted version control (assisted in that we use SVN to help us handle it). Version control is about ensuring that we have several working prototypes of the code, so that we can fall back to them in case we break the code. Another benefit of using version control, is that the other groups will have something to work with, instead of having to wait for that group to finish the entire part of the project.

Part 3.

This part of the mini project will evaluate the choices made in the previous parts of the mini project, based on wheter or not they worked as planned.

First of all, it was decided that an agile approach should be taken to the project, and we decided on using SCRUM. This turned out to be a good solution, as we have benefitted from always knowing when the other groups plan to have certain parts of their projects done, so that we could plan our own development phase accordingly. In the project, we soon discovered that we had planned too many meetings during the sprint, and they were beginning to slow down the work within all the groups in the multi project. As a result of this, we decided to cancel the weekly General Meeting, and chose to merge it with the Sprint Planning Meeting at the beginning of every sprint. This helped the groups increase productivity during the sprints.

When looking at our own use of SCRUM, we quickly discovered that Stand-Up meetings did not work very well with our group, so we decided to replace them with a ”walk and talk” approach, in which the group participants takes a walk while discussing the progress of the project. This served well as a quick reminder of where the project was going, as well as giving the group a well needed breath of fresh air away from the computers.

During our SWOT analysis, we established the basic factors effecting our project productivity, and although most of them were well estimated, there were some exceptions. For instance, we assumed that having the source code from the previous semester would be a great asset to us, which was a wrong assumption, as it turned out to be written in a way that did not allow us to easily reuse it. In fact, we had to rewrite the entire project, with the exception of a few minor functionalities that we could reuse. We only hope that we have written our own project in such a way that the next semester will be able to continue the development instead of having to start over like we did.
In order to improve our project quality, we had decided to use pair programming, refactoring and version control, in addition to the approach described in SCRUM.

It is worth mentioning that we decided on having a SCRUM master in our group. Not because it differs from the techniques described in SCRUM, but because it differs from our usual approach. In the previous projects, we have had no formal way to decide how to distribute our man power and how to settle arguments, which could sometimes be frustrating.
This worked out better than expected, and we had less arguments and heated debates than some of the earlier semestres.

Our first technique, pair programming, had mixed success. Due to unexpected ilnesses, more often than not, only 3 of the four group members were available at the university, which put us in a dilemma: Either we use pair programming, and let the third member work on his own (which seems a bit unfair to the third member), or we split our efforts into 3 groups (more equal), which would decrease the quality of the code. We decided on using a few ad-hoc solutions, such as using pair programming whenever some part of the project turned out to be very difficult to write or complex to understand. If a task seemed very simple, only one man would be assigned to it, and he would then be required to ask for aditional man-power in case the task turned out to be more difficult or complex than previously estimated.

We had decided early on that our project should be written in such a way that the students that will continue on our project will be able to do so without major difficulty, and as such refactoring would be important. We have written comments trhoughout our code to help understanding it, but not always enough, and at the current state, the code does have lines of code that could be refactored and improved. However, we have planned to refactor the code, and we do so gradually. We just don't have a fully refactored and commented product at the current time.

Using subversion control turned out better this year than previous semestres, and because more of a useful tool than a nuisance, which differed greatly from an earlier experience.