Reflection by Bernard

Summary

During the project my contributions mostly consisted of writing the report, discussing new ideas and helping with implementation by pair programming. I also did some occasional testing in the UT3 environment. I took on the role of 'housekeeper'; I checked if code looked good, helped to spot bugs, and worked on the ton of documentation about the project that needed to be updated every week.

I enjoyed the discussions about new functionality and felt like I was part of the group effort. I learned about the way other people write code by pair programming and code review. The set construction of the meetings was good for the development process, because it made the project move forward.

I had to learn how to use new tools for the development process: git, trac, and Eclipse. It was purposeful to learn how to use these effectively, because they will probably be used in future projects. However, GOAL was frustrating to work with at times, because it is basically a list of *if-clauses*. Bugs were hard to find in the GOAL-framework; there is room for improvement here.

In a next project I would be more involved in the implementation of new ideas, instead of evaluating and improving existing code.

Reflection

Here a more general reflection of the project will be described. My own contributions towards the final product and also to organizing the group effort are discussed in greater detail. The decision making process, my roles, what I learned and think can be improved will be reviewed more thoroughly.

Contributions and Roles

During the project my contributions mostly consisted of writing the report, discussing new ideas and helping with implementation by pair programming. I also did some occasional testing in the UT3 environment. In the meetings I had the role of secretary a couple of times.

I enjoyed the discussions about new functionality and felt like I was part of the group effort. After deciding that we wanted to implement a certain idea the group was split up. This was because some group members understood the project's architecture better than others. So my role was often that of 'housekeeper'; I checked if code was understandable, helped to spot bugs, and did work on the ton of documentation about the project that needed to be updated every week. Practically I spend most of my time using Trac or Google Drive. I did some code review every week to keep up with the current state of the project's structure. I liked my role, but I feel that I would have liked an active role in the writing of new code even more. I think this was something that I caused myself, because I did not take the initiative in situations where I should have.

Process

Group projects make you learn how to read other people's code. I had trouble understanding some modules at first. The way other people write code in GOAL is naturally different from how you do it. After more time with the project you start to understand the other group members better. Pair programming helps a lot in this case, it makes you take a look at someone's thought process.

In the previous block I worked on the block's environment, there I did a lot of pair programming too. I feel that GOAL is a language where you benefit from this technique a lot, because

the code needs to be very precise in order to work and there are not built-in checking mechanism in the IDE.

The set construction of the meetings was good for the development process, because it made the project move forward. We first had a small recap of the last week, then we discussed what we were going to do the present week to achieve the new target. The project manual gave a good overview of the general structure of the project. It helped to put everything in perspective.

Improvements

GOAL was frustrating to work with at times, because it is basically a list of *if-clauses*. A common bug was misspelling to variables. Example: 'UnrealID' vs. 'UnrealId'. These bugs cost a lot of time to find, because GOAL does not give an error or a warning. Packaging was not used at the beginning of the project, which resulted in a messy 'workplace'. When packaging was implemented another bug was introduced. The module references were wrong in some cases. The references had to be added manually, they could not be imported like in Java.

Trac had too much functionality for a smaller project like this one. The ticketing system and the wiki did not enhance the project flow, they were just extra documents that had to be update frequently. Trac took more time to keep up-to-date, than that it saved by adding structure. A simpler environment would create sufficient overview of the project.

Future projects

In a next project I would be more involved in the implementation of new ideas, instead of evaluating and improving existing code. I would also put more time in thinking what you are going to code before you are going to code. It sounds logical, but often I just start writing code and see where I end up. Obviously this is not always a good way to go about it.

I would still use mostly the same tools: git, google drive, and eclipse. I have had some time to get used to these development tools. With this experience I can get started quickly on actually writing code in a future project. Although this is more of practical experience it is maybe even more valuable than most of the other things I have learned in this project.