Reflections report

Version: 1.2
Date: 2012-07-25
Authors: Oscar Brodefors, Filip Askviken, Rikard Andersson, Emil Nyström

This version overrides all previous versions.

Table of contents

1. Difference between individual programming and working in a team

This is the first software project for every member in the group where the number of developers are more than two. The biggest difference is the increase of communication paths. In a group of four people, there are 6 possible paths of two-way communication (N*(N-1))/2 = 12/2 = 6) and the risk of misunderstanding in different matters is high. When you are working by yourself, you never need to tell someone else what you have done or what you are doing. This is crucial and critical when working in teams. In order to facilitate the communication work, we have used Google docs (in order for everyone to see the same documents and write documents together) and WhatsApp (communication chat tool for direct and fast communication to everyone in the team). Teamwork also demands better and more commenting in the code, and for this we have used JavaDoc. GitHub has been a major help in the version control and code writing. This project would not be as successful as it is without the help from a shared web-based repository.

All four members in the project have enjoyed the extra challenge of working in a team. You learn a lot when working together with others. The learnings are not only code and programming related, you also learn how other people communicate, solves problems, plans and order working tasks etc. The best thing with working in teams compared to working by yourself is the possibility to get help from others. When you get stuck in a task, you can always ask one of the others for help, and often you can solve the problem together. Especially when you work together in the same room. However, this can also be a disadvantage when you get disturbed of other team members when working. The benefits exceeds the disadvantages, however.

The team has benefited from different backgrounds and members different experiences and skills. Some of us are interested in coding and some of us are more interested in software engineering methodology and management. The different interests increased the quality of the group’s work. United under the common interest in IT, we soon found our group dynamic. Roles were found and tested under the SCRUM method.

The team has been extremely driven and focused on the goals. One quickly realizes how important motivation is for team success and how well teams can perform when there is no need for whip nor carrot from management but rather the team as a whole pushes forward.

Another big difference with working in teams is the fact that you don't need to do everything by yourself. In a efficient and high performing team, all members are motivated to work and have tasks do to. This means that even if you have a busy day with other things, the project could still be going forward because other members are working.

A couple of things we are really proud of in this group, is the fact that we successfully used SCRUM-methodology for this project. Also, we are proud of the fact that we have succeed in working effectively together during the whole project. We stated from the beginning that we wanted to work together, rather than working by ourselves at home. This has enabled better group dynamics, communication, and more effective synergy-effects in the group work.

2. Methods used to create the documents

We have used Google docs in order for everyone to see the same documents and write documents together. Google docs enables commenting on the documents and therefore easy proofreading. Traceability between requirements and test cases was easy to conduct in Google docs. The work was easy to follow for all participants in the project.

A disadvantage with Google docs is the low functionality of version control. Although it is possible to retrieve old versions of a document, it is not very easy or useful. GitHub has a lot of better version control and have also been used in this project.

Documentation have been created continuously during the entire project, the writers working alongside with the coders. Every document have been specified with version, date and authors:

Version: 1.3

Date: 2012-05-17
Authors: Oscar Brodefors, Filip Askviken, Rikard Andersson, Emil Nyström

This version overrides all previous versions.

This enables a good overview of the documents and makes it easy to follow changes and retrieve old documents. The need for version control with regard to documentation writing in this project has been relatively small, but still very useful. The need for retrieving old documents will probably increase as a project grows in number of code lines and complexity. All reversions of the documents have been saved on two different computers in order to retrieve old information when needed. In addition, we also have version control in Google docs as mentioned before. All documents have also been committed to GitHub repository continuously, which also enables extra version control. GitHub has been a major help in the version control and code writing. This project would not be as successful as it is without the help from this web-based shared repository. GitHub has also been useful for uploading weekly hand-in documentation to the course administrators using tags.

We have also used WhatsApp (a communication chat tool for direct and fast communication to everyone in the team) for instant and effective communication. This has been useful when creating the documents since it is easy to ask the others to proofread a document or a piece of text.

3. Appropriateness of the adopted SE techniques

The waterfall method has been used to some extent in this project. The method states that a software project consists of five steps (see figure 1) where one should move to the next step only when its preceding step is completed and perfected.

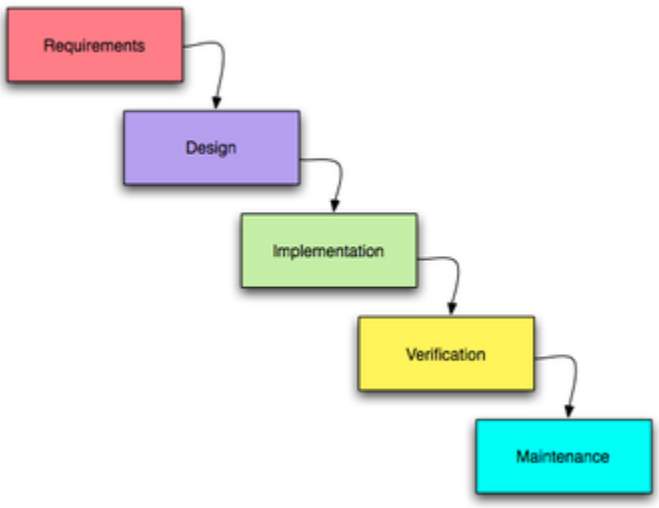


Figure 1 - waterfall method
http://en.wikipedia.org/wiki/File:Waterfall_model_(1).svg

The waterfall process has been useful for this project. It was a good start to write requirements early in the process, which forces you to define the functionality of the application at an early stage. To start with writing requirements helped us to identify some special cases and functionality which we wouldn't have thought of without writing the requirements.

However, the steps in the model above have not been performed in the way the waterfall process states. The steps/phases have been conducted more iterative and not in the strict order presented by the methodology. Some requirements have been changed during the process and some have been added in the implementation process when a need for additional functionality have arisen. This is natural since this is one of the team's first real software projects and the learning curve of processes like writing requirements have been steep. One learning from this project is that it is very hard to write all requirements needed for an application from the beginning. As the project develops, you find new requirements and needed functionality frequently.

A methodology very appropriate and frequently used in this project is Scrum. All members of this development team have enjoyed using and practicing Scrum. It took a couple of weeks in order to get used to the work with sprints, but after that, the methodology has been effective and useful. Everyone knows

what to do and in which order. Scrum has facilitated the prioritization work and enabled a task-driven working process.

Another SE-technique used in this project, to a low extent, is test driven development (TDD). TDD was not supposed to be used in this course and for this project. However, one of tests in this project derived new functionality and therefore, TDD has been tried and used in some extent. This has been interesting and useful according to the developers in this project. One of our internal goal was to practice and learn about different SE-techniques, and using SDD helped us achieving this goal.

In summary, the use of SE-techniques in this project has been interesting and educational. In some extent, the documentation work in the project have been relatively time consuming. Some of the work with the documentation and different techniques, has been over-ambitious for a relatively small software project like this one (for example code coverage). It has however been very educational and interesting to use many of the methods used in large software projects.

Another question raised in discussions within the project team, is the use of many different SE-techniques instead of focusing to 100 percent on implementing one methodology. For a larger project, it might have been of higher importance to focus on less different SE-techniques and instead implement the ones you use to 100 percent. However, for this project, we are satisfied with working with different techniques since it have been very educational.

4. Coverage

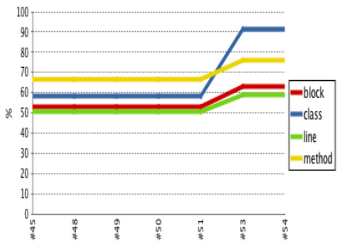
Code coverage is a useful tool within software testing in order to see how much of the code being tested. It is important to remember that it only tells you how much of the code being tested, and not whether your code is tested enough. A code coverage of 100 percent means that all written code is tested. This is of course good, but does not automatically mean you have good testing strategy. From the beginning, we thought 100 percent always would be a optimal goal for code coverage. However, later on we found out that this might not be as easy as it seems. All code cannot be tested in a

good way and writing tests only with the goal of reaching 100 percent code coverage is unnecessary - it is better to focus on writing good tests and use the code coverage rate as a helping tool in order to ensure that most of the code are being tested.

From the beginning, the technique used for code coverage was time consuming and quite difficult to get started with. A lot of time was used trying to get the code coverage software EMMA function with Eclipse in a Android-project. Although, we succeeded with the EMMA-tool using Jenkins. And when the technique was implemented, we found the code coverage tool helpful. The code coverage was helpful in the work of finding areas of the code not exposed to testing. This made us create additional test cases to increase the coverage and robustness of the code. When writing more and more tests, the code coverage percentage increased. To actually see this percentage-rate increase, was found to be very useful in the work of understanding the importance of testing. If you don't get any feedback from your tests and see results when writing them, it is easy to forget the importance of writing tests. It was more fun to write tests when seeing the code coverage rate increase and it became a measure of test quality for us in the group. The code coverage tool definitely helped us develop the UNIT-tests and get an overview of how many UNIT-tests we needed.

For this project, a goal minimum of 30 percent code coverage (lines) was set for the most important classes. It was hard to determine a certain rate of code coverage, but 30 percent seemed appropriate. This goal was reached in the most cases, see picture below.

Package: itBrainiacs.muffins



Coverage Summary

name	class	method	block	line
itBrainiacs.muffins	91,7% 11/12	76,0% 57/75	62,9% 1204/1915	58,7% 310/528

Coverage Breakdown by Source File

name	class	method	block	line
AddBookActivity.java	100,0% 2/2	60,0% 6/10	57,4% 221/385	56,2% 50/88
DataBook.java	100,0% 1/1	89,2% 33/37	86,1% 173/201	86,3% 63/73
DataBookFactory.java	100,0% 1/1	80,0% 4/5	67,9% 391/576	53,3% 106/199
MuffinsActivity.java	100,0% 1/1	100,0% 2/2	100,0% 103/103	100,0% 19/19
MuffinsUtility.java	100,0% 1/1	50,0% 1/2	58,7% 37/63	42,7% 6/15
SearchActivity.java	100,0% 1/1	100,0% 4/4	100,0% 130/130	100,0% 25/25
SearchItemActivity.java	0,0% 0/1	0,0% 0/2	0,0% 0/4	0,0% 0/2
SearchResultActivity.java	100,0% 2/2	75,0% 3/4	59,8% 55/92	73,7% 14/19
ServerCommunicator.java	100,0% 1/1	50,0% 2/4	73,0% 65/89	60,0% 17/28
SettingsActivity.java	100,0% 1/1	40,0% 2/5	10,7% 29/272	16,7% 10/60

The rate of 30 percent can be seen as low and the goal needs to be revised progressively. Already now, we can see that a higher rate of code coverage is reached and therefore we can increase the goal in order to ensure testing quality for the project. When we discussed the level of a new percentage goal in our project team, we disagreed and had problems of finding arguments for a specific level. After some discussion however, we could agree on a new goal:

We want to aim for an increasing trend in code coverage

Junior programmers like us should not focus on a certain level of code coverage. It it hard and demands a lot of experience. It is better if we focus on writing good tests and use code coverage as a complementary tool for ensuring that a major part of the code is being tested; aiming for a increasing trend in coverage.

Another, unexpected, benefit from using code coverage was found in the project. The code coverage tool visualized redundant test cases when the percentage rate did not increase after writing new tests. This could be useful in a very large project.

If we would have conducted a larger project with higher complexity, the code coverage work (and formulating good tests) would have been of even higher importance.

5. General comments about the project as a whole

The members of this team have enjoyed the practice in software development widely. It have been fun to both code a Android application and practice different software engineering techniques. Using waterfall method, Scrum, GitHub, code coverage - EMMA, nightly builds - Jenkins etcetera has been very educative and useful. Although, a lot of time has been used in order to get the techniques and different tools to work. It would have been good if more time could have been placed on using the tools, instead of installing the tools.

One of the hardest parts within software development is the time planning and how long time a specific implementation takes. This has been present in this project which gave us useful experience for the future.

Working with this Android-project have also been interesting due to the future potential and hype regarding application development for smartphones. This project has made us familiar with all major tools for developing applications, which has been very attractive and appealing. This course will definitely be useful for us in the future!