Personal learning goals

Context project: A Search and Rescue Mission

June 25, 2014

On this page, all team members will reflect on their personal learning goals, which they made in the beginning of the project.

Learning goals

Wendy

My learning goals for this project were:

- Learn to work with LaTeX
- Learn to work with GitHub
- Gain as much project-experience as possible, as I am going to need it next year for my minor

I think I reached my learning goals with success:

- I am able to make simple documents in Latex now. A team member told me about ShareLaTex and I think it is really handy, I even used it outside this project already.
- I am finally able to work with GitHub without tons of conflicts everywhere. I worked with GitHub before but because of all the problems I got, I was a bit afraid to use it again. With this project I was forced to use GitHub again and I am glad I had to. Now I can pull, commit and merge without problems and I even learned to work with Git Shell.
- I also gained a lot of project-experience. We used a lot of handy project tools (like Trello, Google Spreadsheets, GitHub) and because we were with so many (19 students) a lot of organization was necessary.

Except from this learning goals, I also learned other things which I did not think of beforehand:

- I learned unit testing.
- My java programming got way, way better than before the project.
- I executed my first user tests.

Joop

When starting the project we all put some goals for ourselves to be reached at the end of the project. My learning goals were as follows

- Improve testing abilities
- Acquire project manager experience and skills
- Have a clearer view of how projects are done.

I am happy to say that I am satisfied with what I have achieved and learned during the project. When I started with the project I wanted to be the project manager, since this role interests me and I wanted to figure out whether this is a role I may want to put through in my carreer. During the project I have had several meetings with the project coordinator where we discussed the role and how I could improve it. I think this was a great experience for me, especially because we were in such a large group.

Although my role as project manager took a fair amount of time I did manage to improve my testing abilities. I wrote multiple tests for different classes and methods, which in my eyes has given me a clear understanding of how tests should be used and what its use is.

Due to my project manager role I got a pretty clear overview of how projects work. I have had contact with the customers, context leaders, context coordinators and the team. I am now familiar with the Scrum framework, which taught me that there are many missteps to be made during a project. Overall I think I can say I accomplished all of my learning goals.

Katia

My learning goals at the start of the project were:

1. Understand an existing code base and modify/extend it.
2. Working together as a big group.
3. How to use git(hub).

I have achieved all of the goals I set for myself, plus some additional things:

1. During the project, I mostly worked on adding new features. To add the features we had to modify the existing code base, for which I had to look at the code to understand how to modify it.
2. In my team, communicating with each other was not a problem. If someone needed help or finished all of their tasks early, they would just indicate it. Using Scrum also helped in that regard. I have learnt that communication, especially in a group as big as this one, is really important. I also didn't always have a good idea of what the other teams were doing, mostly due to poor communication.

3. Before this project I didn't have much experience with git(hub). I have now learnt how to do the basic operations with git, and also how and why to use branches and pull requests.

4. During the project we used Scrum to manage our product development. Previously I have only heard descriptions of how Scrum works, but I have never actually applied it myself. By using it in the project, I have gained a better understanding of the Scrum framework. I found that the Scrum meetings were really helpful during the process, because they made it easier for everyone in my team to understand where we were at. Because of that, we could easily determine if we would be able to finish our tasks by the end of the week.

5. I have learnt that unit tests are important, especially in a big project as this one. Some of my functionality got lost during the merges because some of the tests I wrote didn't cover enough of the functionality. I have now learnt how to write better, more helpful and useful tests.

6. Even though I have already had previous experience with LaTeX, my knowledge has been expanded by using it to write some documentation.

Arun

My learning goals:

- Learning to work in a big development group
- Learning to understand legacy code en add features to it
- Learn how to use github
- Learn how to make a product according to the customers wishes

What I learned in this project:

- I learned how to work in a bigger development group than I am used too. The group consisted of 19 people and with a group this big it is important to communicate with the group what you are doing and going to do otherwise it could create problems.
- I learned that the best way to add features to legacy code is to first go over the legacy code and make sure you understand it first before adding features. It helps to have a uml diagram or something like that to understand legacy code
- I learned how to work with github and now find this a very useful tool for software development.
- I learned that when creating software for a customer it is important to update the customer frequently on what is happening and that the customer agrees with these changes. It is also important to make sure the customer knows what they want.

I succeeded with all my learning goals.

Seu Man

The learning goals I had set initially before starting the project have been reached. I believe I have learned even more.

During this project, working in a large group is what I have learned the most. Never before have I worked in such a large group of 19 students. I have learned how important communication and making agreements is.

This is also the first time that I worked on an existing software system by improving and extending it. For me the challenge was understanding how the system is build up and extending it. My programming skills have improved too and I also fully grasp the MVC structure.

Throughout the project we made use of Scrum. I have learned doing daily stand-ups, sprint plannings, sprint reviews and sprint reflections. We also used timeboxing a couple of times, which was useful.

Using GitHub was completely new to me, but it went pretty smoothly. I now know how to handle minor conflicts. I also know how to make pull-requests and a whole lot more.

Improving my LaTeXskills was also one of my learning goals. I have to a certain extent improved it. Working with LaTeXfeels more natural now. At the beginning I had to look up the commands in google. Now that I have worked with it quite frequently the past few weeks, I do not have to look up the commands that much anymore.

Further I have also learned using handy tools like Trello and Jenkins. The most important thing I have learned from Information Skills is how to list references the right way in a report. And through Project Skills I have learned about entitativity and how the group process is influenced by it.

All in all, I have learned a handful of project skills and tools for me to take to the following years.

Nick

My learning goals consisted of learning more about Latex, Git(Hub) and JUnit. I achieved all of these goals.

- At the start of the project, when we still worked a bit more on the essays, I learned quite a bit about Latex, for example about the nature of Latex. So that when you download Latex, you still can't do much since it can only be used for compiling. You also have to download one of the available editors. Furthermore I learned how I could combine multiple .tex files into one essay, how I could make a glossary and some more miscellaneous stuff within Latex.
- My second learning goal was to learn to use Git(Hub). This worked out very well. I have learned to work with the basic operation of

Git (such as clone, fetch, merge, add, commit and push), but also with operations to go back to an earlier state of the code, or to ignore certain files while committing. Furthermore I learned about branching and how it can be done best with such a big team (and what can go wrong when you don't do it this way). I've also seen that while working in big teams it is very inconvenient to have too many branches, since all these branches have to be merged with the master branch which can take up a lot of time, especially when these branches have been around for quite some time without having been merged. GitHub itself was pretty straightforward, so I learned what it had to offer quickly and that was that.

- JUnit was pretty easy to learn, this was also because I was enrolled in a course called Software Quality and Testing that required me to know much about this. I also witnessed how a JUnit test found a failure in a piece of code that I had written and that was pretty nice to see. Unfortunately this scenario only arose once and in the other cases when JUnit complained about a failing tests, it wasn't about a fault in my code at all, but it had to with the test not expecting my new feature/change. This was quite annoying, since I had to then rewrite the failing test which took extra time. Because of this I formed the opinion that JUnit tests covering small pieces (units) of the code only take extra time to write, but barely have any revenue. What I did think of as useful were the integration tests. That is giving the system an input and verifying that the output stays the same after changes made to the code. That the system changes on the inside and therefor breaks some unit tests, isn't really important I think, as long as the final desired output is generated properly.

Sander

- Working in a big group.
- Usage of GitHub and branches.
- Working with dependencies between parts of software.
- The practical working of a Client-Server Architecture.
- Using Scrum and estimating working loads and times.
- Learn to tell group members what is troubling me about them (and their jobs).

  My list of learning goals this project is achieved.
  The working in a big group was a big challenge. The meetings with other groups were often very short and didn't give me with the information I wanted. Therefore I needed to take a good look at the code of the other groups to know exactly was has been done.
  The intensive use of GitHub and Git has had a really positive outcome. I know a lot more about Git than I imagined beforehand,

which is positive as I am using Git at my current job as well.
I have not worked a lot on the dependencies between software parts. I know how to add them now, and how to make use of them in a Maven project, but that is it. It would be great to learn how the exact link between software parts works and what happens if the dependencies were added in e.g. another place.
This project was a great example of how a Client-Server Architecture is built. Although it was not ideal, I know the practical working of the architecture and how the communication between the components is done.
Estimating working loads and working times turned out to be a complicated job as well. As our job was refactoring, it was hard to find a job on which the estimation could be reflected. Using Scrum was nice though. I noticed, especially in such a big group, that communication is the key to a good product. You have to communicate about what you have done, are going to do and what you are doing right now. This makes sure no work is done twice and no complications between software parts are found afterwards.
Telling others what is troubling me about them and their jobs was a hard task in the beginning. Jan and Martin are experienced programmers, that was noticed real quick as they moved at the speed of light. They also worked on code and jobs of others(including me), and that bothered me. I wanted to learn from this project as well and the project is not meant to let work done by the ones who already know how to do jobs.
Telling this in a group made some fraction in the beginning, but after a good meeting and talk everyone understood each other and we could work as a good team. Martin and Jan gave a helping hand where needed and did not work on others jobs anymore without consultation with the group members.
Afterwards I am glad I had chosen for refactoring, as I learned a lot about coding itself. I watch the quality of my own code much more when writing it, in stead of afterwards.

Tim

I have surely learned how people work in teams on large software systems. This is the first time that I work together with a team larger than five people, and the first time that I work on a software system that is going to be used after we are done with it. The scale of the project required us to use Scrum as a development method, so I surely have learned how someone can use Scrum to develop large software systems within a relatively short amount of time. I have also learned how to use multiple design and test principles that were covered in the SQT and SEM courses, so I fulfilled that learning goal as well. Learning to work with Git was mainly achieved by trying to demolish

Git as much as possible at the start of the project, so the learning goal concerning that is achieved as well. Only the goal of optimizing my communicative skills remains, and I didn't spend a lot of time on achieving this goal. I, however, think my communicative skills are alright as is: I was the leader of my group during the MAS project of the first year, and I have mainly practiced my communicative skills there. This goal was, therefore, less important to me than the other goals during this project.

Joost

From the beginning of the project I had the following learning goals that I wanted to achieve:

- Learn to work in an effective way within a large group.
- Learn about the different roles within a Scrum team and what it is like to take the lead as Scrum Master.
- Learn to use GitHub.
- Learn more about different testing techniques and continuous integration.
- Learn how to understand the needs of the client in a way that works for both the client and the development team.

I believe that I have learned a lot about these different topics and have achieved most (if not all) of them.

- Being in a group of 19 students was a real challenge itself. However, during the project we have truly seen that our productivity within the group has risen enormously. I believe that this is something none of us has ever dealt with and we have succeeded in together.
- At the beginning I was not sure about what the role exactly meant and what my responsibilites were. I have never had such a role in a team so it took me some time to get into this. The main role of the scrum master could be described as to make the scrum team work as efficiently as possible and we have seen that throughout the project our efficiency has almost doubled. I now know what it is like to take this role and will be glad to take it on me in a next project because although it started a little rough I feel comfortable doing it now.
- Before this project I had never used GitHub but during the project we have used GitHub for all integration amongst every individual. This way I have learned to use it and although I am not yet an expert I do feel comfortable using it while it scared me a bit at first.
- During the project I have done a lot of testing on our code. This was a great way for me to put skills we learn in a different

course into practice. I have also learned a lot about continuous integration and the absolute necessity of it in large a project.
- There was a lot of discussion about the features that we were going to implement in this project and the final product that we were going to deliver. Because the client was not always consistent in their needs it was sometimes a challenge for us to know and to verify what we were doing. Later in the project this communication became better and better as we as a group became better in communicating with the client and passing its needs on to the development teams.

Sille

- Learn how to work with Git in a large group
- Properly understand the SCRUM process, and how to lead a development team as scrum master
- Improve and/or rewrite overly complex code using design patterns

My main goals for this project were:

- Learn how to work with Git in a large group.
- Properly understand the SCRUM process, and how to lead a development team as scrum master
- Improve and/or rewrite overly complex code using design patterns

Looking back, I can happily say that I archieved these goals. However, there's always room for improvement. Working with Git has been fun and I can now comfortably stage, commit, push, revert and merge with git. Learning by doing was quite effective for archieving this goal. I can't say I know all the ins and outs of Git now, but this project established a firm basis.

I'm glad we had to strictly use the SCRUM process for this project. While I think my performance as scrum master wasn't bad, but I definitely could have kept more oversight of the process and especially the documents that were submitted. Part of this lack in oversight was the illogical and forced sprint periods, starting on friday and ending on thursday.

The usage of design patterns went very well. I found a piece of code that needed a complete overhaul, and that could be done with a combination of design patterns I didn't know of before. It was very satisfying to see the much improved result pass all the tests.

Shirley

My learning goals were:

- Learning to work with architectures, like client-server, and understanding how they work.
- Working with already existing code. So understanding how a system works en why it works the way it does.
- Programming in the wild, with this I mean programming without a clear assignment and without lectures where you can find what to do and in which direction you need to search.
- Programming/working in a bigger project. So far I only have programmed in small projects, in pairs or individual.
- Using Git
- Apply Scrum

What I have learned this project:

I have learned a lot during this project, either with coding as with the process.
Firstly, I have learned to work with an architecture consciously. I had learned about the theory behind architecture already, but never put them in practice. Now I understand better how to apply architectures.

At first I though that it was difficult to use architectures, but basically it is just following a set of rules. Its all about how classes communicate with each other and how they depend on each other.
Working with the already existing code was really challenging. There was a lot of code and it was hard to figure out how the classes where connected. In some situations, this resulted in searching for something in the code for quite a while. You knew that something exist in the code but you didnt remember were you had seen it.
During this process I learned to use some very useful functionalities of eclipse, for example searching through your entire workspace. What was also very helpful was making and using a UML diagram. It gave a good overview about how classes where connected.

Programming in the wild, is something I did not do much. This because with refactoring, most of the time you were rewrite old code. So you don not produce a lot of new code and new functionalities. However I did learned that this wild programming is not that wild after all. You do not have a teacher who gives you a specified assignment, but you do have to give it yourself. You need to know what you are doing and what your goal is.

Working in a project of this size was something really instructive. I

already knew that communication is very important, but this project really emphasized this. It is good to know what someone else is doing and how issues were tackled. Coding with multiple persons on the same piece of code was challenging. In some situations this went really well, but it also happened that there were a lot of conflicts during the merging process. This brings me to the next point, the use of git(hub). Git(hub) is definitely something I would use in another project. It supported the merging process very well. Also the fact that you could see who worked on/modified a file is great. I learned how to work with branches quite well. It gives you as programmer the change just to try something without being afraid of failing. If something did not work, you just do not commit/merge it.

The last learning goal was to apply Scrum. In the beginning this was quite hard, especially making the sprint plans. As the project progressed, it became easier to define the tasks that needed to be done during the sprint. Something that did not go well was assigning storypoints, especially assigning them for one sprint. The next time that I will use Scrum, I would keep up how many points there were assigned in one sprint and how many were actually done. Another thing I would do different, is the division of the sprints. I would prefer to have a sprint who starts at monday instead of friday. I also would prefer to make a two week sprint, instead on one (unless working full time on a project).

Because of this project I now pay more attention to maintainability and extensibility of code. One of the most important things I have learned, code-wise, is that I need to doubt less about how I would attack a problem but just need to try things. I am really glad I have chosen to join the refactoring team.

I have learned things I otherwise never had learned. Despite the fact that the refactoring was learning-full and nice to do, it did confirm that it is not something I would like to do in the future.
For me personally, this really helps me making better choices in the future like my master choice next year.

Tom

My learning goals were as follows:

- Working together in a large group
- Learning to use GitHub
- More JUnit experience
- Learning to work with Scrum

For the first point, while I didn't quite get much experience working with the full group, I believe it's quite accurate anyhow as nobody wants to work in groups that large. It is only natural to split them up, and even the split up group of 7 was the biggest group I've ever worked in.
So even though I didn't get experience working in a group as large as I originally expected, in the end I do believe it was a valuable experience.

My goal for GitHub was definitely achieved.
Combined with the experience I got through the Software Quality and Testing course and the Context Project I'm quite used to GitHub as a whole now, both the GUI and command line versions.
In particular this project was very useful for learning to coordinate and combine work on Git thanks to the group being very large.

As for JUnit experience, this was more than achieved.
I've spent a lot of time writing tests and used various ways to implement them. As a result I can quickly identify which approach (Standard tests, Parameterized, Mocking, etc.) is best to use and am experienced at using them.
If anything I somewhat regret not being able to earn some experience with Power Mocks (mocking of static classes) because of its complexity. Still, the knowledge it exists will prove useful in the future.

Learning to work with Scrum is also definitely achieved.
Every week we learned new things, learned to improve our workflow and the amount of story points that can be completed.
I would've liked to have the start and end of Sprints at a different day though. I believe ending on Monday and starting on Tuesday would be much better as it'd allow us to better utilise the weekend.
In addition to that, due to the short length of the project, by the time we became comfortable with the Sprints the project was already at an end which I felt was a pity.
Regardless, it was a great learning experience to work in a well-defined workflow.

Finally, this project also shows clearly why you need to be careful when structuring your code to maintain a simple and clear design, how to do so and how to solve it for existing codebases. Overall, the project was a valuable learning experience.

Valentine

- How to work together as a team.

- How to use Design Patterns.
- How to apply Software Engineering principles to a project.
- How to make a better use of GitHub.

Have I reached my learning goals? Yes.

- I learned how to work together as a team, and to be honest - never again! At least not in a team with a considerable amount of people. It was fun working in small groups, but the coordination between teams was poorly done and a lot of misunderstandings originated from that problem; we were just too many.
- I also learned about how to apply Software Engineering principles in real life, and how to use different design patterns that exist. Design patterns such as MVC for GUI's or Decorator Patterns are beautiful concepts and enjoyable to implement, but when faced with big-sized UI's or giant classes to decorate, things tend to get tough.
- I really like going through the Scrum process. It is well organized, and very nice to take part in. There were some issues, though, because we are still "beginners". Maybe with a little more experience, the project's process could have gone a bit better.
- I have now learned how to make proper use of GitHub thanks to this project. I see now how handy it can be. I will use it in projects of my own from now on.

Xander

At the start of the project I decided to become Scrum Master, that way I had to be involved in situation in which my two main weaknesses of working in a group would be challenged.

- Learn to cope with the ever changing situation in a SCRUM team
  Scrum itself forces you to be flexible in what you do and want. Every week is a completely new start of another part of the project and everything can change in the blink of an eye. I hate it to not be able to plan ahead and know what is expected from me. As a Scrum Master I was forced to work closely with Joop (our product Owner), this meant being involved in the backlog and customer meetings. During the projects I started to get the a grip on the situation. Unfortunately the programming phase of the project was only 5 weeks, therefore I have not really been able to develop this skill. A longer project would have been much better and allowed me to really work on this skill.
- Learn to cope with people that don't exactly know what they want/say

I hate it when people have a strong opinion about something without exactly knowing what is going on. During a project this big it is inevitable that people say and want things even though they are not sure if it is possible. As a Scrum Master I was constantly involved in conversations concerning the new features and problems arising within the teams. Especially in the beginning of the project I noticed my opinion was to fierce and group members did not really know what to do with it. During the project I tried to be milder in my opinion.

Ruben

My learning goals were to better work together in a large group, I wanted to learn how to work with Github and also how to realise a project using Scrum. I have achieved all of these learning goals.

- To learn to work together in a large group is a fairly general learning goal. What it mostly came down to was that you had take other groups than just your own into account. We only sat at the same table with your own sub group. So you had to maintain communication between groups. In the beginning of the project this was all a little bit difficult. Also because you had to think about division of the workload and that this did not collide. It cannot be that one group has to wait on another because they have not finished their task yet.
- Before this project I had not worked with Github. Ik had an abstract idea of branches in which you built a small feature and when it was done you merged it with the main branch. But that was about it. During this project I really learned to work with it and really saw its usefulness. What is also very neat about Github is the overview of your project and all the branches you get.
- My last learning goal was how you realise your project using Scrum. In the beginning it was not fully clear to me how to go about using Scrum. Meetings started quite slow because I, but also others, did not exactly knew what had to be discussed. Luckily we quickly came into some sort of flow and it all went much smoother after that. The purpose of Scrum is to divide your time into smaller blocks, like weeks. The flow we got in was a result of this so in my opinion it was successful. .

Calvin

- LaTeX
- Teamwork within a group ('company')
- Scrum

- Architecture design and implementation
- Unit Testing

The first goal I reached was improving my latex knowledge. Prior to working on the project I've dabbled with latex before, but never properly investigated how it worked. Usually my files were full of warnings, overfull or underfull boxes, and other issues. Luckily a member of my group had quite a bit of latex experience. From asking him certain questions and analyzing his latex documents I came to understand what I was doing wrong, and how these mistakes and issues can be mitigated.

My second goal, teamwork within a group, has also been reached. I really enjoyed the composition and organization of the team. The smaller groups within the team simulated a department within a company. I really liked this since it meant we'd have to learn how to communicate with the other groups (departments). Especially the creation of the Scenario Editor was a great learning experience, as we had to coordinate the work happening in two different groups. And because it didn't always go that smoothly I also gained valuable experience in detecting possible issues which could impede the teamwork.

A project requirement was the use of Scrum and Trello for managing the process. Before the context project I've never had to use Scrum. While skeptical at first, in the end I couldn't have been happier with Scrum. The plans created using the Scrum method were usually completed at the end of the week, and I never felt as if I did too much or too little. Trello was also very handy to have, it makes keeping track of progress very easy. In fact, I enjoyed using Trello so much that I am currently using it in two separate projects as well.

The one goal I did not fully achieve was Architecture design and implementation. Our group (group 3) was mostly concerned with the scenario GUI, the human player GUI and in the end, the path planner. While we had to design and implement the Scenario GUI as it was made from scratch, I wasn't the same level of design and implementation I had in mind. I was mostly thinking about the refactoring that had to happen to BW4T, which group 1 did. But in the end I think I fit better in group 3 due to the fact that group 1 consisted mostly of TA's and I am sure that my second learning goal would have not been completed if I were in that group, since they had less contact time.

Finally, unit testing! It may seem weird to have it as a goal since one of the requirements for the context project is the successful completion of the course Software Quality & Testing. However, after that course, I never got a chance to use my acquired knowledge in practice. It was fairly complicated for me when I started writing unit tests, especially since there were so many things to test, and our first

project had a fairly large GUI which also had to be tested. But during the course of the project I slowly learned the best practices for testing.

Martin

Initial learning goals:

- How to set up a well organized testing environment with Jenkins, Maven and Sonar.
- How to manage a large project consisting out of many teammembers in a versioning-system like Git.
- Working in a team with the SCRUM method.
- Unit testing and creating power mocks in order to test static system classes.
- Communicating within a team and cooperate together with a client to create a smooth transition at project delivery.

Reflection:
I more than exceeded the learning goals which I had set initially. I managed to create a very nice setup with Jenkins, Maven and Sonar. Which initially took quite some time to set up. I managed to create a multi-module Maven project, which gets automatically pulled from the Git repository and then built and tested on the Jenkins server. Afterwards Jenkins runs Sonar and reports the code quality of the project.

I learned how to manage a large project on GitHub, but I also learned that for future large project, it will be a lot easier if you use software like GitLab. The reason for this is that you can specify permissions for groups of users. This way users can't change files which really shouldn't be changed. GitHub has the feature, but you should pay for it, to be able to use it.

Working in a team with the SCRUM method is very rewarding, but I have seen that it is very important for the scrum master to keep track of the work that is being performed in the group and set strict deadlines to the group members. Otherwise tasks just don't get done.

I basically gave up on creating power mocks to test static system classes, because it's nonsense with all the different bugs and different java versions for the systems which are available now(easymoc or mockito).

What I learned about communication with the client is, at the beginning of the project make a stakeholder analysis and create a matrix in which you specify each and every stakeholder and set the priority as to how you will priority handeling tasks with the client. If you do not do this, you can grossly get lost in tasks, which actually should not be performed.

Jan

- How to work together in a team on the same project commiting to the same files.
- How to refactor a large project, specifically how to decide what are necessary changes.
- Applying the SCRUM Method.
- How to organize a team as a Scrum Master.
- How to help people without doing the work for them.

During the project I learned different methods of how to work together with multiple people on the same files. When programming it is very easy to integrate the work from different people using a versioning system, in our case we used git in conjunction with the website github which acts as central point to upload the most recent changes to. When working with documents containing tables or text there are multiple options. If the document is a report it is easiest written in LaTeXwhich can be compiled into many different file formats. When working with excel type files using google docs is the easiest, as it allows multiple people to change the document in real-time.

I made a lot of progress in understanding which problems to fix when faced with refactoring a large code base. For instance it is very important to do your changes in the correct order. To rewrite parts of a class without reorganizing the general structure first does not make sense. First test to check the working state of the programm need to be written, then the organization of the project should be improved then the individual problems need to be asserted and last of all you should work on documenting the changes.

Working in a SCRUM team has been a very challenging experience, there are many subtle problems that can creep up to you. For instance when the scrum master does not show enough leadership the team will stagnate and not get anything done, as tasks are just postponed to the next and the next week. A lot of time gets lost with what should be short tasks.

I was able to get the experience of being a scrum master for a week as our scrum master went on vacation. I was quite happy doing the extra management work and I felt that we were making more progress than in the weeks before as I made sure that team member knew they just needed to ask if they were stuck on a problem instead of trying to work it out for weeks on end.

I did get a little further on trying to get people to find solutions to problems they ask me about. The strategy I implored was to ask what they had thought up so far and then tried to lead their thinking to a solution by asking more in-depth questions. Basically not directly telling them the answer but leading their thoughts through

a correct path of thinking. This method proved quite successful for most questions I got, but some of them I did not have an answer myself so I included them in my search, so they could see and adopt the strategies for searching something.

Daniel

- **Documentatation/LaTeX**: I got the role of 'documentator' of the group to practise my documentation skills. I was really able to learn latex and I'm already using it for other projects/documents outside the project.
- **GIT**: I have learned to work with *GIT* in terms of using a repostory. Really setting up my own project in *GIT* is something I'll have to learn another time.
- **Maintenance**: We used different systems to be able to maintain the software. I learned to work with *Sonar* and *Maven*. Those programs show where there are possible problems and how many tests work, what is covered etc. Also I learned how to set up checkstyles in eclipse for java.
- **scrum**: It was the first time a really applied scrum in a project. Throughout the project I learned a lot of things to improve teamwork and to create a stable development system. Review a lot of things and work more efficient.
- **Coding**: While I was in the refactor team I had to understand the code and 'dive' into it. I learned how to search through code and how to create a better understanding for myself. I didn't learn to write 'new code' but did write a lot of test code what i hadn't done before.