Final Report

OBJECT ORIENTED PROGRAMMING PROJECT

Jari van den Broek, Johannes Hagspiel, Nathan Ordonez, Nick van Riet, Lukas Zimmerhackl, Ardy Zwanenburg | CSE1105 TU DELFT | 04-04-2019 GROUP 22

Table of Contents

General

This project was all about improving our knowledge of the Java programming language and seeing what it's like to work in a team, in this case consisting of six team member, and a client. The goal was to build an application in which users would be able to track how much CO_2 they have saved and also see how much their friends have saved.

THE PLANNING

The planning was done via the Scrum framework[1]. Scrum allowed us to set clear goals for each week and gave us a clear overview of what every member was responsible for and what still had to be done. After the weekly meetings on Monday, we assigned new tasks via the Scrumboard in our GitLab[2]. Every week we would make a Backlog and a Retrospective. The Backlog would contain the stories from the client and the tasks we assigned to our team members and the priority of the given task. The Retrospective would contain the problems we encountered during a certain week and how we would adjust to them to improve ourselves in the future.

THE COLLABORATION

Our collaboration went very smoothly. Everyone was committed to making every deadline and delivering a product that was solid. There were no major issues and everyone did their best, which resulted in a great working environment where everyone could speak their minds and voice their ideas, which greatly benefitted our creativity and thereby also the end-product.

THE COMMUNICATION

The communication was mostly done during our weekly meetings on Monday and our group chat in WhatsApp. All our communication was in English so that everyone would be able to understand each other at any given point. If at any point we found it helpful to do so, we would also meet during other days of the week to discuss our work and help one another out with their issues. The communication went well and we always made sure everyone was involved in the conversation and got the chance to speak their minds.

USING GITLAB

GitLab was one of the most difficult parts about this project, but as we progressed throughout the project our understanding of its functions and the proper usage grew a lot. We did make some mistakes at the start of the project, such as committing directly to the masters, but with the help of our TA, we got better and better at using Git. At the later parts of our project, the biggest challenge was keeping our GitLab clean and our branches to a minimum.

---

[1] 'Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.' – Scrum. What is Scrum?. Retrieved 03/04/2019, from https://www.scrum.org/resources/what-is-scrum

[2] 'Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.' – Git. Git distributed even if your workflow isn't. retrieved 03/04/2019, from https://git-scm.com/

WHAT DID WE LEARN AS A GROUP

The biggest takeaways from this project are the proper usage of Git, the better understanding of programming and how to use Scrum to manage your project.
The usage of version control will be helpful in every project to come, and the same is true for the Scrum framework, which will help us in all our future projects, no matter what the subject is. The greater understanding of Java and, more in general, programming itself will benefit us greatly during our studies and will form a great foundation to build upon.

Design decisions

WORK DIVISION

At the start of the project, we divided the project into several parts and assigned them to groups of people. The GUI was assigned to Ardy and Nick. They first worked on the login screen and after that created the second screen with the main application. Lukas assisted them whenever they needed him to and made the CO_2 numbers work. The server side was assigned to Johannes, Nathan, and Jari. At first, they worked on making a server using telnet and combining that with a MySQL database. After a while, we switched to SPRING, which Johannes implemented with some help of Nathan and Jari. Nathan also set up a Raspberry Pi for continuous deployment. Making the backlog and retrospective for the Scrum aspects of the project was mostly done by Johannes, Nathan, and Jari at the start and at the end it was mostly made by Lukas. Making the Agenda was usually done by Nick. We all did our part in making sure everything was uploaded before every deadline. The Final Report and Presentation were mostly made by Jari.

TECHNOLOGICAL DECISIONS

The Server

In the development journey of our project, we implemented the server in two different ways: once based on telnet and then based on the SPRING framework[3]. Even though we achieved great results with our earlier version, our TA pointed out that we had to implement a REST API[4], which was not possible while using telnet. Thus, we had to look for a second solution, where we landed on the SPRING framework. Making use of the tools provided by various SPRING projects such as SPRING Web MVC, we created a REST API. For the servlet container, we used Apache Tomcat and the underlying database was a MYSQL database. Additionally, we

[3] The SPRING Framework is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application' – Wikipedia. Spring Framework. retrieved 03/04/2019 from, https://en.wikipedia.org/wiki/Spring_Framework

[4] 'Representational State Transfer (REST) is a software architectural style that defines a set of constraints to be used for creating Web services ' – Wikipedia. Representational state transfer. retrieved 03/04/2019 from, https://en.wikipedia.org/wiki/Representational_state_transfer

used hibernate for object-relational mapping[5]. The reason for these design decisions is that they were made following an in-depth tutorial that explained in detail how to best make use of the SPRING framework in the context of a REST API. Thus, what spoke for making use of these tools is that they were accompanied by a guide, that clearly laid out the strengths and the weaknesses of these tools as well as how to best implement them. Obviously, other tools were also considered, such as using a Postgres database. However, after researching the general consensus in the programming community as to the usefulness of all these tools and as well as after having taken into consideration the fact that already a working prototype was created, the decision was made to stick with these services. Looking back, the group unanimously would reaffirm this choice.

Client-Server Communication

As stated previously, based on the TAs feedback, client-server communication had to be implemented via a REST API and via JSON[6]. The endpoints of the API were set up using the Web section of the SPRING framework and the parsing of POJO's[7] into JSON and back was achieved through the Jackson. In detail, the client would send, depending on the goal to be achieved, a PUT, GET, POST or DELETE request to a specific URL where the server then accepts this particular request, reads the body and then performs the appropriate action. Again, similarly to the server, this process was implemented by following a tutorial for the REST API and again, other alternatives were taken into consideration. However, after having seen that this particular way of implementing a REST API seems to be the standard, we decided to stick to it. Looking back at this decision, it is clear that this particular implementation provided us with the flexibility to add new and change already existing request endpoints without a lot of trouble. Conclusively, we no doubt would go down this path again.

The Graphical User Interface

For the development of the GUI, we started with the login screen. We mainly used the Eclipse WindowBuilder plugin to shape this screen, and afterward wrote code to add some basic functionality, including: Login in, creating an account, and reading from and writing to file. We eventually ended up using a database instead of a file, but the fundament remained. The main screen was made in a similar way: roughly shaping it and adding all buttons needed. After all major functionality was implemented, we started with giving the GUI a better design using Swing elements with the help of the WindowBuilder Plugin. We made a lot of changes, including the recoloring of everything, resizing and repositioning buttons, and using a different letter type. This resulted in a GUI that looks a lot better than the standard design, and absolutely magnificent knowing this is fully done in Java.
Furthermore, we also made some small decisions to enhance the user experience. Think about features like opening a pop up when creating an account, opening your profile page and adding friends. There also went a considerable amount of time in the implementation of the leaderboard. A major obstacle was adding images, as this seems to work in Eclipse (although it was very inflexible), but not in IntelliJ. After a long struggle, we finally succeeded. This breakthrough allowed us to add stimuli (including achievements and the growing tree).

---

[5] Object-relational mapping is a programming technique that is used to convert data between incompatible type systems using object-oriented programming.

[6] Java Script Object Notation.

[7] Plain Old Java Object.

Point for improvement

SOFTWARE IMPROVEMENT

The biggest improvements to our application could be making it into an actual Android application. The features that we have implemented would also work very well on an android app and it would be more convenient for the user to be able to say they saved CO_2 on the fly instead of using their computer. Another improvement would be to have a choice of which car the user would normally drive so that the estimated CO_2 can be calculated even more precise. Something else to look into would be using the login and achievements from Facebook, not only would this add more incentive for the user but it would also guarantee the safety of password and username combination.

PROCESS IMPROVEMENT

Our planning and time management is something that could definitely be improved. We often delayed our work until Sunday and then we started working on it, instead of doing it as soon as possible. On the week after our demo's we would often not be very productive as there was no pressure of a deadline the next week. By setting clear goals and making sure everyone did their part every week we could improve our process a lot. We used the Scrumboard for this, but if we had put more emphasis on it, our process would have been better and our workloads would have been distributed more evenly over the weeks.

COURSE IMPROVEMENT

The biggest improvements that should be made for next year would be the explanation of GitLab and giving us more good practice rules from the start. We made quite a lot of errors concerning Git that could have easily been avoided if that information had been more clear from the start. Another improvement would be more feedback from the TA on how many points we earned after a demo and if the team is doing well enough to get the points for subjects such as Scrum and the overall process. A more detailed rubric would also make a fine addition to this course, as the current rubric seems to miss some important information.

Ethics

During this project, we aimed to design for effectiveness and environment-mindfulness. Our main goal was to make an application that clearly showed the users their CO_2 emission for certain (daily) activities and how much they can save by choosing greener alternatives. One other value that we valued highly was public accountability. By making the user able to see, not only their own but also the CO_2 consumption of other people, we hope that users will be more inclined to choose the better option. The clear overview of how much you are doing for or against the environment will make people more aware of the impact they are having, which they would normally have never accounted for. A by-product of the way that we are promoting

the greener alternative could also mean that people would choose vegetarian meals more often, which could lead to a higher demand of vegetarian food in supermarkets and consequently for a lower demand of the more environmentally-tasking meat products. We wanted to design for those who want to be more aware of how much they are doing for or against the environment, so that they can be mindful of the environment and see clear ways on how to lessen their CO_2 emissions. One of the things that would change during our project if we were solely focusing on the values we were designing for, would be giving more information on how people were affecting the environment by giving tips and explanations on why, for example, a car is so much worse than using the public transport. Another more interactive and personal way to enforce the environmental mindfulness, is to confront people by giving them weekly report containing their emissions.

To gain more insight into the values and stakeholders for whom we are designing we would consult an expert in a field very different than ours. One of those options would be an expert in Psychology. Someone who understands the way people can react to our design decisions could greatly change the way we have been designing our project if our ideas do not match reality. Does the sort of 'peer pressure' of being able to see one another's scores truly improve overall scores, or does it demotivate people because they see that their contributions are too small to be noticed anyway? Is the way that we are presenting the amount saved and the amount still used the best way for making the users aware of how much they are doing for the environment, how do such numbers affect them? These are points that such an expert could help us with.

We believe that we successfully designed our application to improve the environment-mindfulness of the users by clearly showing what people can do to lower their emissions by giving them several features in which they can choose between the bad and the better. The way that our point system works is based on how much CO_2 certain things emit or save, that way it gives a good representation of one's footprint in the real world. Another thing we have tried to design for was public accountability. By giving people the option to become friends with one another and see scores of their friends, we hope to increase the amount people save because the user should not want to be lacking in comparison to their friends. The ability to talk to each other via the global chat can be a way for one user to motivate another to choose the greener alternative.

Individual Feedback

JARI VAN DEN BROEK

Strong points

I learned a lot about programming in Java and programming in general. There were a lot of different ways to use Java which were really great to discover and see which works best for our purposes. I think my strongest point in this project was communication. Being active in all conversations and brainstorming with everyone and seeing what was still needed and doing that was great.

Weak points

One of my weaker points was the difficulty of the things I coded. If I compare the level of my work to some of my peers, I think mine was a bit sub-par. Even though I learned quite a lot about the Java programming language, some of my teammates excelled at one subject while I stayed quite mediocre in all of them. Another weak point was my planning, a lot of the times I

had to cram all of the work on the Sunday before the deadline instead of doing it over multiple days.

Conflicts

There were no conflicts within the team, but that could also have been because there was a clear difference between how vocal everyone was. All of us improved our knowledge of Java, but not everyone improved their communicative skills during the project, which was a bit disappointing. Overall it was a good process and I was very content with my teammates.

JOHANNES HAGSPIEL

Strong points

Overall, I did learn a lot about Java programming, mainly about the SPRING framework. I think my stronger points were that I worked independently and on a self-motivated basis, which resulted in me creating on my own the server and the part of the client that is responsible for the communication with the server.

Weak points

I think one of my weaker points was that I was not really able to explain how the server and the client-server communication worked. I feel particularly disappointed about this as I anticipated this and even told my teammates about it – I mentioned it as a weakness of mine in the README. Additionally, I did not really lead our team and push it during the periods when not much progress was being made, which was also partially caused by the point I mentioned earlier, my inability to teach other people about things that I have learned.

Conflicts

As far as I remember, there were no major conflicts in the team. Even though on the surface, this looks good, I do think that this is also a sign of the fact that we did not engage too deeply into the areas that other people were responsible for. Overall, I think I had great teammates, I learned a lot about the SPRING framework and the importance as well as the difficulty of project management were highlighted again to me.

NATHAN ORDONEZ

Strong points

Making sure that deliverables were done on time, noticing the small details and deciding how features were going to get implemented, as well as learning how Maven, JaCoCo, Checkstyle, Git and Gitlab work in general. I think I helped well when people were having trouble with those. Also getting ideas for bonus points, setting up a physical server to run the code we made. I think I made progress in communication in the sense that I communicated my work and gave my honest opinions about other people's work (which was my goal in the README), and I think I helped the group work together to a certain extent when it came to

coordinating our works and distributing workloads. Also helping write meaningful scrum backlogs and retrospectives, writing the meeting notes in an easy-to-read format.

Weak points

I didn't code as much as I should have, and I think I let my teammates take on work that I should have taken on on my own. Overall I think I made mistakes in planning, and in judging the priority of this course/project compared to the other ones.

Conflicts

No conflicts really, I like my team and I think if all of us were working in a company together (no exams), we would have done something actually marketable.

NICK VAN RIET

Strong points

I started this project eager to learn new things concerning Java applications. As a result, I've learned a lot from my teammates. I've been able to broaden my knowledge on GUI design and functionality, server-client communication, testing (and automated tests), and the general creation of a Java program from scratch. The project also enabled me to improve on some communicative skills: Working with teammates, and using a version control program (GitLab).

Weak points

In hindsight, one of my weaker points was that I lacked flexibility and mainly worked on a single task (the implementation of the GUI). While this resulted in a single task done right, it also accounted for the fact that I wasn't able to help my teammates that much with their own problems. Furthermore, when I was struggling (for example, with the implementation of certain features or with technical difficulty), I didn't go to my teammates for help as often as I should, resulting in precious time lost. I'm aware of what I could have done better (also concerning feedback from teammates), and I will surely try to improve on these weak points.

Conflicts

As far as I know, there weren't any major conflicts within our team. During the project we often assigned tasks and everyone was (more or less) able to complete this assigned task on time and with satisfactory quality. The only noteworthy point concerning conflicts is that we, as a team, sometimes had the tendency to work on the weekly necessities at the very last moment. This often resulted in a very frustrating Sunday afternoon. But nonetheless, we were able to succeed to meet all deadlines, and I firmly believe we can be proud of what all our hard work resulted in.

LUKAS ZIMMERHACKL

Strong points

For me, my strong points were sticking to a schedule. I attended all the group sessions, apart from the one when I was sick, and did my best to make the deadlines. I think I improved my scheduling a lot. I also learned a lot about working in a group and what I should do during group sessions, even though they are still beneath average. I also learned a lot about how to work with Git, Maven, Checkstyle and IntelliJ.

Weak points

One of my weak points was input. I found it difficult to contribute to the conversation during meetings and I have been trying to improve that by speaking up more. I didn't expect that to be so difficult. I also should have taken more initiative. I did all the tasks that my team members gave me but I didn't think of many tasks myself which led to me having a low number of commits. I also should have asked the TA for help earlier with my technical difficulties. I coded less interesting things because of this than my peers which led me to have beneath average coding input.

Conflicts

I don't think we had any big conflicts as a team and I think this shows that we were all on the same page and that nobody in our team was intentionally doing something wrong. I didn't speak up enough during the beginning of our project and in the week 5 peer review thing everyone said I should speak up more and that has helped me with trying to improve my communication skills. I don't have any big problems with any of my teammates.

ARDY ZWANENBURG

Strong points

After this project, I can proudly say: "I survived Git". After struggling with understanding what Git is, how it works and how to use it, it has proven to me that Git is a good tool for team projects and I am really happy that I now can say that Git is one of my stronger points. Still can improve on it by committing more and have a better look at others their merge requests, but that can still be learned in further projects. Starting this project I also learned a lot about JFrames. How to create a JFrame and implement functionality to it. And to end one of my stronger points is now getting a problem before me and just work in a solving process to solve the problem and also look further than just solving it. Also making sure that your solution does not give any problems somewhere later in time.

Weak points

I still think communication is still one of my weak points. I am a person that is more of a planner and makes sure that he can go to bed before a deadline instead of working on it until one minute before the deadline and in this group there are more people who are more saving their work until the last moment and that gives conflicts. Especially when I need something from someone else or someone needs my help, but saves it for the last moment just before the deadline. I think I should be more aware of this and sometimes stay up later for those questions. It is also sometimes hard to stay of others work. Then I see nothing is happening and ask if they can do it and end up doing it myself because I was already working on a different part of it.

Conflicts

There weren't any really big or nasty conflicts in our group. The only things were discussions over small implementations. That for example I changed a constructor because it was easier to test, but turned out in someone else his code to start giving problems without really knowing why.