MILESTONE 3.2: IMPLEMENTATION

343.309, UE Software Engineering

Winter-Semester 2020

033 521

TOPIC

Entire System Implementation

Group Nr.2:

- **Omar Dueñas:** Subsystem II
K51849509, roqueluisd@me.com
- **Michael Lengauer**: Subsystem I
K11710940, lengauer.michael@gmail.com
- **Mario Lischka**: Subsystem III Traffic Flow
K11712790, Mario.lischka@gmx.at
- **Eduard-Florin Hogea**, Subsystem IV
K12019919, eduard.hogea00@e-uvt.ro
- **Lukas Wais**: Subsystem III Security
K11816105, lukas.wais@outlook.de

1. Reflection on our code!

a. What would you make different to improve the usability of your APIs/interfaces/GUIs?

Subsystem 1 Michael Lengauer

To improve the usability of the APIs I would use JASON objects instead of XML and Plain Text as I implemented it now. Furthermore, would a graphical representation of the Map be helpful to get a quick overlook over eventual problems.

Subsystem 2 Omar Dueñas

Maybe better coordination with the other subsystem would have been better for the usability. Although we worked very closely together in the previous tasks, which leads to the final implementation, the way of coding and technologies are not the same. This caused some problems in the final implementation.

Subsystem 3

Mario Lischka

Not just to improve the usability of all APIs doPost requests should be implemented. Also, to serve with better security. Because everyone can change things, if he has access to the OperatorAPI.

Lukas Wais

I would improve the connection with the other subsystems. To get a more automated information flow. Moreover, I would provide some graphical user interface to test interface and provide statistics and general information about the running system.

Subsystem 4 Eduard-Florin Hogea

The way that the subsystems communicate with each other makes the whole application not that robust. I think that a different approach may have improved the overall stability of it and with that the usability would have improved also.

b. What was the hardest part to implement and why?

Subsystem 1 Michael Lengauer

The hardest part to implement where the REST APIs as the technology stack used was quite new to me. Also getting the Subprojects to work together not only in theory but letting them run on a single Tomcat instance proved impossible due to miss communication of the used versions.

Subsystem 2 Omar Dueñas

The hardest part of the implementation was the definition, how the routes are going to look like and request them. Also, the “illustration” of the route for a better understanding of the users.

Subsystem 3

Mario Lischka

The hardest part to implement was the pathfinding, due to the decision of designing the map like a grid. Thus, I was not able to use some predefined search algorithm (e.g. Dijkstra).

Lukas Wais

The hardest parts to implement were the security scan and the backup since they hardly relay on third party software and actual hardware (backup).

Subsystem 4 Eduard-Florin Hogea

The hardest part of implementing was making it work with the other subsystems. Tomcat was our choice but making it work for all of subsystems was not an easy task. Probably one of the reasons was lack of experience using it in this specific way.

c. What technological decision do you regret or are happy with?

Subsystem 1 Michael Lengauer

I'm happy that we used Tomcat for our Servlet hosting as it is a easy to work with, stable and small formfactor Application server.

Subsystem 2 Omar Dueñas

One of the best decisions at the beginning of the project was to create a template for the implementation. This help a lot, since the structure was given. Of course, everyone has a unique way of coding and this leads to "issues" in the end. But I believe without structure, could have been more.
Since there was not a lot of time to compare other technologies and probably due to a lack of knowledge, there are no regrets regarding the tools we used.

Subsystem 3

Lukas Wais

I am happy to choose Java and a stateless RestAPI. It is scalable and once upset straightforward to program and extend software. The documentation is also great.

Regret: a simpler server with not so much config would be nice. Tomcat is very big and if you do not have any experience with it, hard to set up.

Subsystem 4 Eduard-Florin Hogea

I am happy with the decision of splitting the application in different subsystems, it lets each on of us focus more on his specific part but the downside of that we had to use things like Tomcat to make it work and the lack of experience in working with it made it harder than it needed to be.

2. Reflection on the previously made design.

d. How well could you fulfill your found requirements?

Subsystem 1 Michael Lengauer

Some of the original requirement where not defined with the scope of implementation and where to big to fulfil. The selected requirements for implementation where more easy to achieve.

Subsystem 2 Omar Dueñas

I could achieve every requirement which is not dependent on other third-party solutions or hardware.

Subsystem 3

Mario Lischka

I was able to fulfill the requirements I chose as features.

Lukas Wais

I could achieve every requirement which is not dependent on other third-party solutions or hardware.

Subsystem 4 Eduard-Florin Hogea

The specific part of implementing the code of what the subsystem needs to do was done without trouble.

e. How well did your design match your implementation and where and why are there differences?

Subsystem 1 Michael Lengauer

I was able to meet the basic functionality design quite well, some additional functionality like a GUI where not implemented due to time restrictions. Also some designed classes turned out to be unnecessary in the implementation.

Subsystem 2 Omar Dueñas

In the previous milestones, we already had to design (with the UML Diagrams) how the systems should work. Therefore, I tried to strictly follow the already design structure in the diagrams. Nevertheless, it turned out, that I had some mistakes in my design and implemented there some pieces from other subsystems. These parts were left out in the actual programming.

Subsystem 3

Mario Lischka

The UserInterface Requirements are just some requirements I added to make clear what should be in the final product.

Lukas Wais

By designing the implementation, I had the actual software in mind, thus it is very similar. I implemented it like the class and UML diagrams.

Differences were made by some classes, like technicians. There are some data structures to provide some sample data. Moreover, I used data encapsulation. In the actual code are now getter and setter methods. There is also a nested class in Security Scan to have an easier and more readable implementation of the scan information.

Subsystem 4 Eduard-Florin Hogea

The implementation followed the design, so it was not that much of a difference. But the initial designs were either unimplementable or just wrong and needed changes.

f. Which requirements were hard to grasp?

Subsystem 1 Michael Lengauer

All requirements focused on quality or improvement where due to the limited scope of the implementation not fulfillable.

Subsystem 2 Omar Dueñas

All the requirements with dependencies from other subsystems.

Subsystem 3

Mario Lischka

It was hard to define the boundaries of the ControlSystem. The design did change almost every milestone. Until the start of the implementation it was not safe how the classes would look like.

Lukas Wais

Basically, everything that dealt with time, or hard to measure requirements.

Subsystem 4 Eduard-Florin Hogea

Making them subsystem to work together.