Logbook for supplementary examination

Information:

The document includes a summary about what everyone in the group have done at the same time it includes the Examiner's remarks with a short summary after every point with what we done about it and who worked on it.

Working log:

Erik

- I have abstracted away the file format of saved/loaded messages by creating an Interface in which the implementation decides the file format rather than having it hardcoded as a csv file in the client model. This was done as two separate interfaces, one for loading previous messages and one for saving new ones.

- I have looked over the SDD and made a few clarifications, mainly in the client section 2.1.3 and server section 2.1.5 (section numbering according to the final report), I have not added further illustrations as I feel the Improved domain model will clarify the inner workings of the project visually.

- I have worked on the Observers, however my changes were later scrapped due to a superior and more complete solution. I tried standardizing the "update" method to where it no longer needed to send a message with it, however we later went with dividing the Observer into a separate Interface for the Client and Server.

Mohamad

- Declared a new interface that specifies the operations on a voice message. The implemntation of the new interface replaces he old concrete implemntations (PlaybackThread and RecordThread). Separate code that works on voice files and move it from the controller to the new implentation.

- Added A new properties file that holds the directory paths used by the application.It holds also other project specific configurations such as the audio fomats.

- The configurations is loaded automatically when the program starts(loadProperties()).

- Unit Testing: testing the output service class required mocking dependencies such as the client model and the objectoutputstream, injecting the dependencies, capturing arguments and finally using byte arrays to serialize the captured arguments.

- Modularize code by separating the IVoice to FileServices package from remote services.

Carl

- I added the designmodel to the repository, if changes are made to the model please update the designmodel here --> https://www.lucidchart.com/documents/edit/cde24b83-a94c-4d47-8fc4-807ec6fbff61/0?shared=true to reflect the changes and I will ensure that the final version is up on sunday 18th

- Exception handling has been overhauled for most of the project.

  Turns out we were throwing a lot of unnecessary exceptions for no reason; these have been removed. We handle relevant exceptions by either exiting the program (if we cannot recover), by prompting the user about the error or by recovering (try again, cancel some connection etc..)

- I updated the domain model to more closely represent the most important parts of the project.

  One could argue that "sockets" are too low-level to be in the domain model, but since they are such a central technology we worked around it makes it in to the model. Other minor features such as file handling and audiorecording have not been put in the domain model as we did not see it fit there.

Yazan

- I Worked on "- File paths are hard coded", "- The address of the server is hard coded":

- added a setting dropdown menu with "change server ip" and "choose history file path" in order to abstract away the file paths and the server address and did their functionality.

- Created Configurations class with its interface and abstracted the functionality of getting from and saving to the config.properties file to methods in this class which I made as a Singelton so it's been instructed in the beginning of the program controlling the

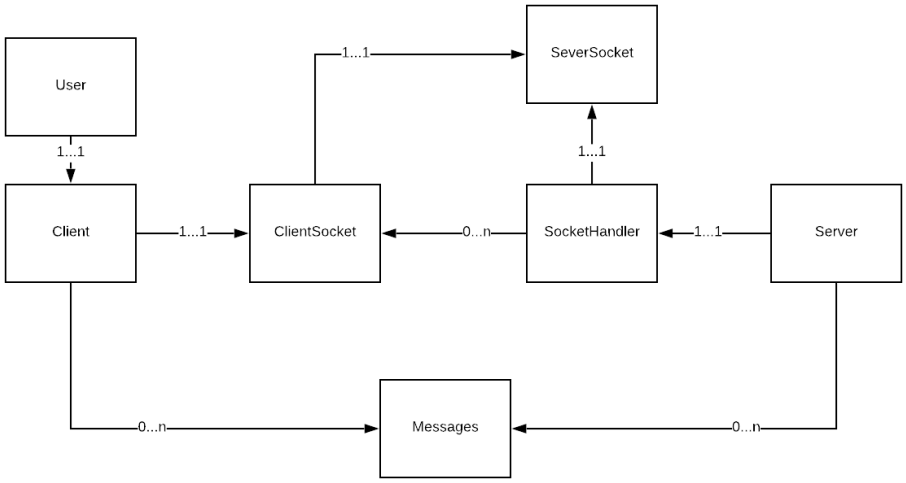
configuration properties file and the other componants of the program using the same instance to read and write from/to the configuration file, hiding the configuration file to be directly accessed.

- Modularized code by separating the services package to RemoteServices which taking care of connecting to the server and reading/writing from/to it, and FileServices which has everything with file functionality to do with.

- Worked on the observers, added notify to observable, deleted disconnect from observable and specified the observers by separating the observable/observer interfaces from the lib to the client and the server so every project has its own observer/observable interfaces following interface segregation principle the abstract methods in the interfaces has different semantic and may be extended in a different way between the client project and server project.

The Examiner's remarks - a short summary – further information in the RAD, SDD and the other submitted files.

- The domain model is too simple and does not describe the actual model. Carl Östling



- The design model is missing altogether. Yazan Ghafir – Mohamad Almasri

- The design-model is in the following link in both pdf and jpg format:

https://drive.google.com/drive/folders/1s1HoZjDztcwHtBJruFObtM9bDYbE8ssg?usp=sharing

- You defined only one unit test for the client, despite that it has the most code. - add unit-tests for both server and client. Mohamad Almasri

- The code coverage for the tests is too low. Mohamad worked

- The SDD does not describe in enough detail how the application works, it lacks some illustrations that explain how it works. Erik Gunnarsson

- Exceptions are mostly ignored (print just stack traces). Carl Östling - Yazan Ghafir

- Not defined specialized observers (general observers do not reveal the intent). Carl Östling - Yazan Ghafir – Erik Gunnarsson

-File related functionality in the (client) model. Mohamad Almasri – Yazan Ghafir

- File format is not abstracted away. Erik Gunnarsson

- File paths are hard coded. Yazan Ghafir – Mohamad Almasri

- The address of the server is hard coded. Yazan Ghafir - Mohamad Almasri

- The meeting notes were hard to follow, difficult to see who did what, and in many different formats. All of us – made it clear who worked on which point, with summary about what everyone did.

If you want, you can take a re-examination in the next omtentamen period. You need to improve your project - Some improvement point we have worked on:

- Added Singelton design pattern to Configurations class so the same instance controlling the properties file is handled by other classes,
- morularized the services package to RemoteServices and FileServices,
- applied interface segregation principle in ISaveMessages and ILoadMessages there they otherwise could be in one interface with unused methods in such implementer class,

- We have worked on Dependency inversion principle, with abstracting the concrete implementation of many classes such as all of the services package's classes by interfaces, and made the other classes depending on the abstractions (the interfaces).
- and separation of concern where we separated the file utilities from the model and remoteServices.

and address all the points mentioned above All mentioned points has now been addressed.