Chapter 1

Introduction

Chapter 2

Technical review

Chapter 3

Requirements specifications

3.0.1 Usecases

This project handle login as sessions. Therefore each session (a set of iterations) is a new user with a new password. However, all these "users" are of the same type and have the same functionnalities :
Mainly, the purpose of this software is to solve an equation with different given parameters and to display the result graphically.

3.0.2 Functional and non functional requirements

On the previous usecase, different functionalities are written. This subsection will detailed those functionalities into functional and non functional requirements.

Functional requirements

ID	Requirement
1	To log into a session
2	To create a new session
3	As logged in : To enter new inputs
4	As logged in : To start or stop iterating on the equation (exchange of data with the cluster)
5	As logged in : To display the result of the iterations in real time on a graph
6	As logged in : To display the result of the iterations in real time as a wing modeling
7	As logged in : To download the log file (exchange of data with the cluster)
8	As logged in : To logout from a session

Non-functional requirements

Type	Requirement
Security	To log into a session with a password so only people having those information can access to it
Reliability	To have a recovery system to access previous steps of a session
Speed	To use a cluster for solving the equation

3.0.3 Further information

An example has been given for the software user interface. With the functionalities required, the final product should have two "pages", one for the connexion or creation of a session, and another one once logged in a session, as the mockups below:

3.1 System design

3.1.1 Component diagram

We began the design process by dividing the system into bigger modules and designing the interfaces between them to conduce parallel work. We split the software into 4 bigger modules and assigned one to each team member:

1. Database & Session: Marion Le Guével
2. Atlas Connector: Ádám Koleszár
3. GUI: Jacek Czyrnek
4. Wing Visualizer: Mihály Lengyel

The following diagram shows the components of the system and the interfaces they interact through.

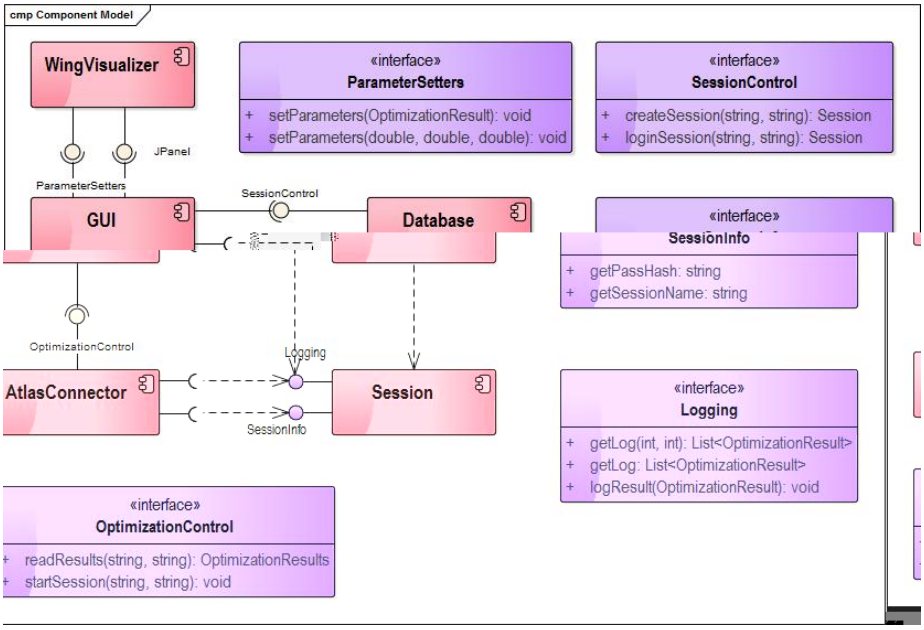

Figure 3.1: Component diagram

3.1.2 Class diagram

In the following diagram you can see, that the classes in the implemented system are simple reflections of the component design above.

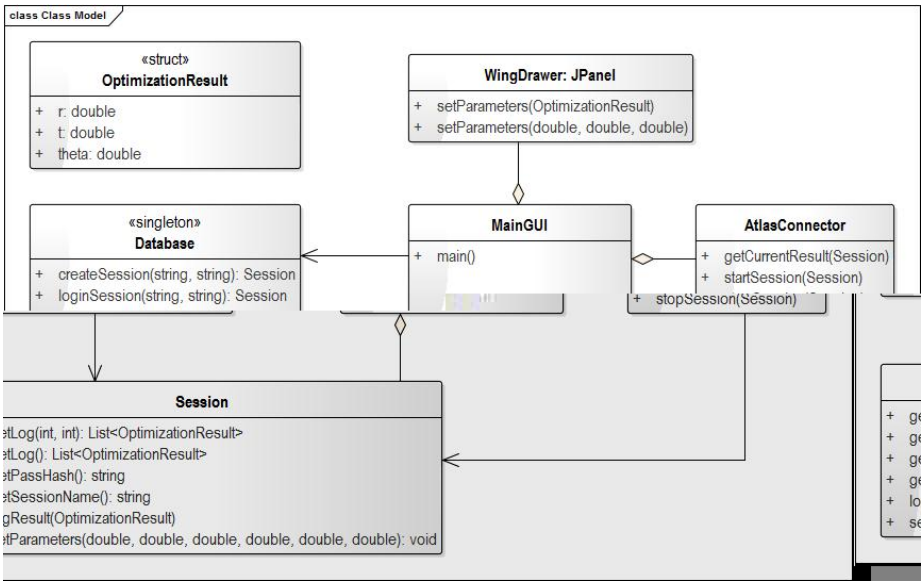

Figure 3.2: Class diagram

3.1.3 Sequence diagram

To complement the above static diagram here we provide a diagram to show the dynamic behaviour of the system. This is not an exact sequence diagram but it shows the flow of data through the system.

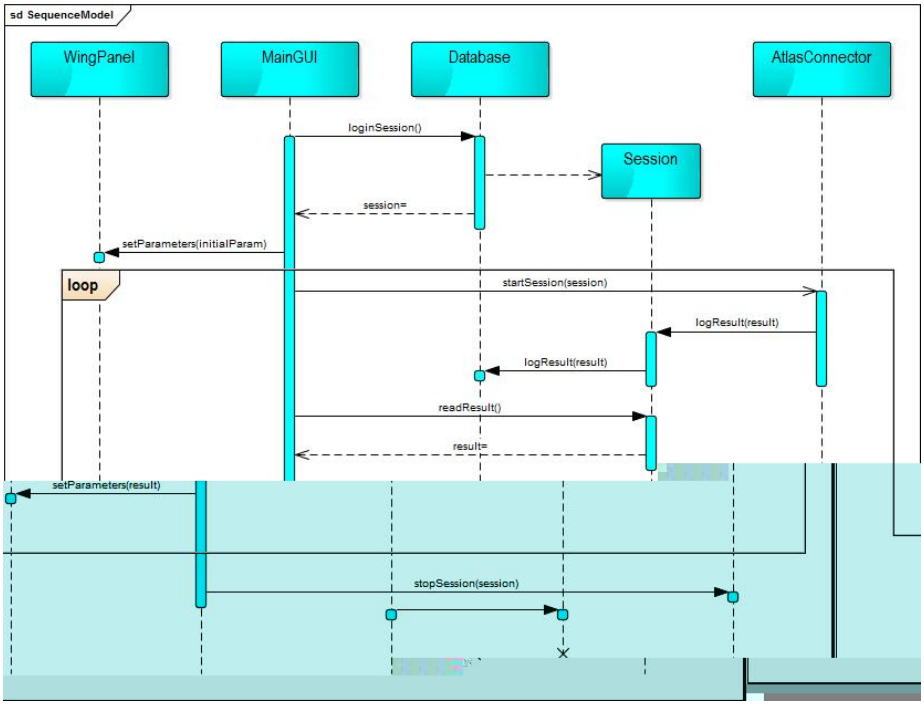

Figure 3.3: Sequence diagram