Design Document

Jens Henninger Daniel Maier Paul Fink
Florian Jennewein

December 1, 2015

Contents

I Architectural Design **1**

1 Introduction **3**

2 External Layer **5**

3 Structural Layer **7**
3.1 Basic 7
3.2 Client 7
3.2.1 Server 10

4 Unused Patterns **13**

5 Interaction Layer **15**

Part I

Architectural Design

Chapter 1

Introduction

This architectural document describes the architecture of the desired system "Sojabohne" from different perspectives and in different levels of detail. All references to requirements are based on the system requirements of our requirements-document from 26.11.2015[1].

[1] https://moodle.uni-mainz.de/pluginfile.php/33185/assignsubmission_file/submission_files/8942/Abgabe_Review_Document.tar.gz

Chapter 2

External Layer

Because there is no connection between our system and other ones, Sojabohne is set in a standalone-context(cf.Figure 2.1).

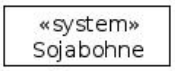

Figure 2.1: Our system in standalone context :)

Chapter 3

Structural Layer

3.1 Basic

Obviously the most basic systemstructure is based on the Client-Server Pattern. That's because in this pattern, the server provides services, in our case mainly the execution of data mining and machine learning algorithms on a user uploaded data. An user shall be able to access the result and the data itself from all over the world(if he has the required rights to do so) via the web application(cf. NFR017.0/NFR005.0). So we use the client-server-pattern, where the web application is the client and the server is the system, who executes the algorithms inclusive the database(cf. Fig. 3.1).

Figure 3.1: Web-application as client; with the server.

3.2 Client

For the client structure we use the layer-pattern, to ensure the requested modularity (cf. NFR026.0). So every layer can be changed, updated or replaced without any complications as long as the new components are based on the right interfaces.
We split our client in three sections: the presentation-, the logic- and the data section.
The highest level is the presentation; the only part of the web-application, a normal user can interact with. There are three surfaces needed for the whole presentation: the user-(for FR005.0, Fr009 1./2.), the upload/download(for the

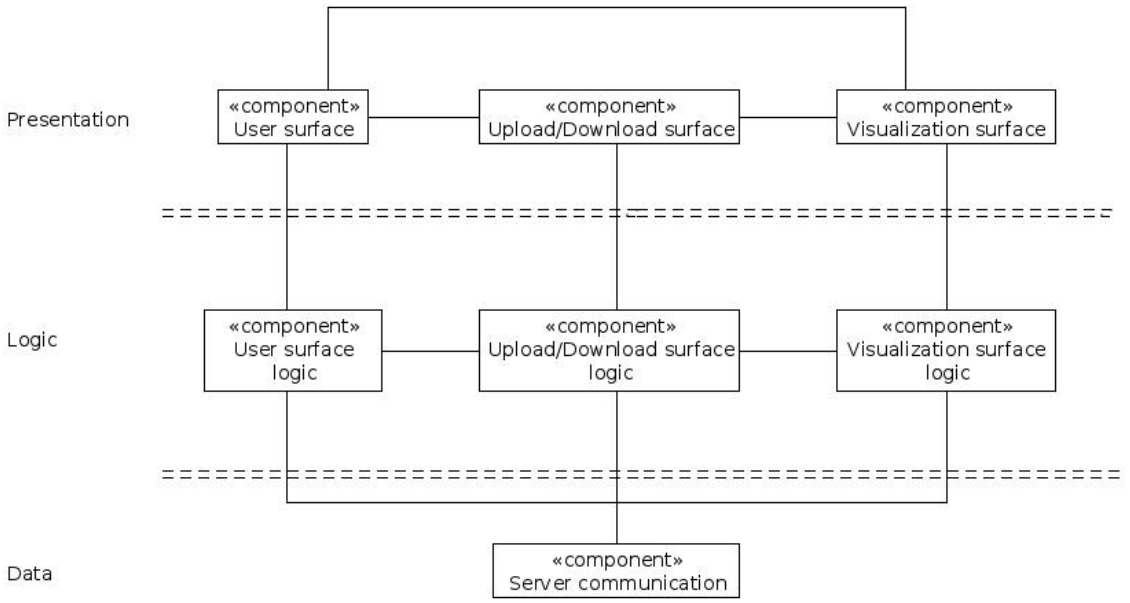

Figure 3.2: Client overview

rest of FR009) surface for everything regarding up- and downloding data and the visulalization-(FR009 3. without the up- and downloading) for the result and data representation and . The posibility to switch between this surfaces is representated by the connections between them(cf. Fig 3.2).
The components in the logic section are used to control the computer operations standing behind the surfaces.
The only component in the data section is used for communication/connection between the web- application and the server, because there's no data stored or processed in the web-app itself.
In our case we have a clean seperation between our layers, because each level can only interact only with neighbouring levels, where each communication is triggered by the highest level. Next we look at the first two sections in greater detail. First the user-perspective and its logical component:
This perspective can be divided into two parts.
First a registration/login component(cf. FR005) and second an administrative component (e.g. changing permissions on data packages; cf. FR009 1./2.).
To realize those components we used model-view-controller-pattern, which has

three parts: model, view and controller. In our case this means for example that when a user tries to login to our system he/she enters the data into the login-view, which informs the login-controller. The controller now communicates with the logic level, which contains the login-model. The model sends a request to the data level and through it to the database to check the given information for its correctness. After receiving that information the model informs the controller, which relays the it in readable form for the user to the login-view. That's in our opinion the most effective way to ensure user permissions(by checking his given data (e.g. password)). Next to the user perspective

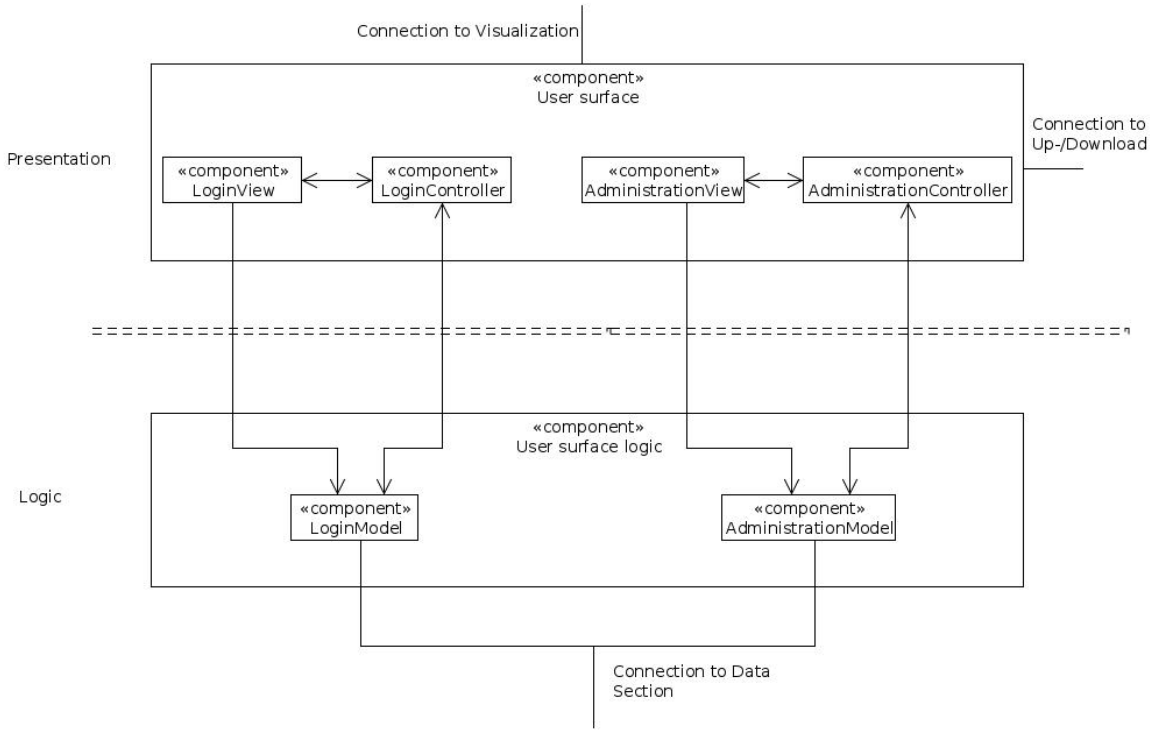

Figure 3.3: User surface zoom

we have the up- and download perspective. Which we can obviously divide into an up- and a download component. We used the model-view-controller pattern portray these components(for permission checking and modularity). Additionally there is a converter, which turns the output into a user chosen format (e.g. pdf). For the same reasons we used this pattern for the visualization surface. Here, in our opinion is the most important advantage the modularity of result

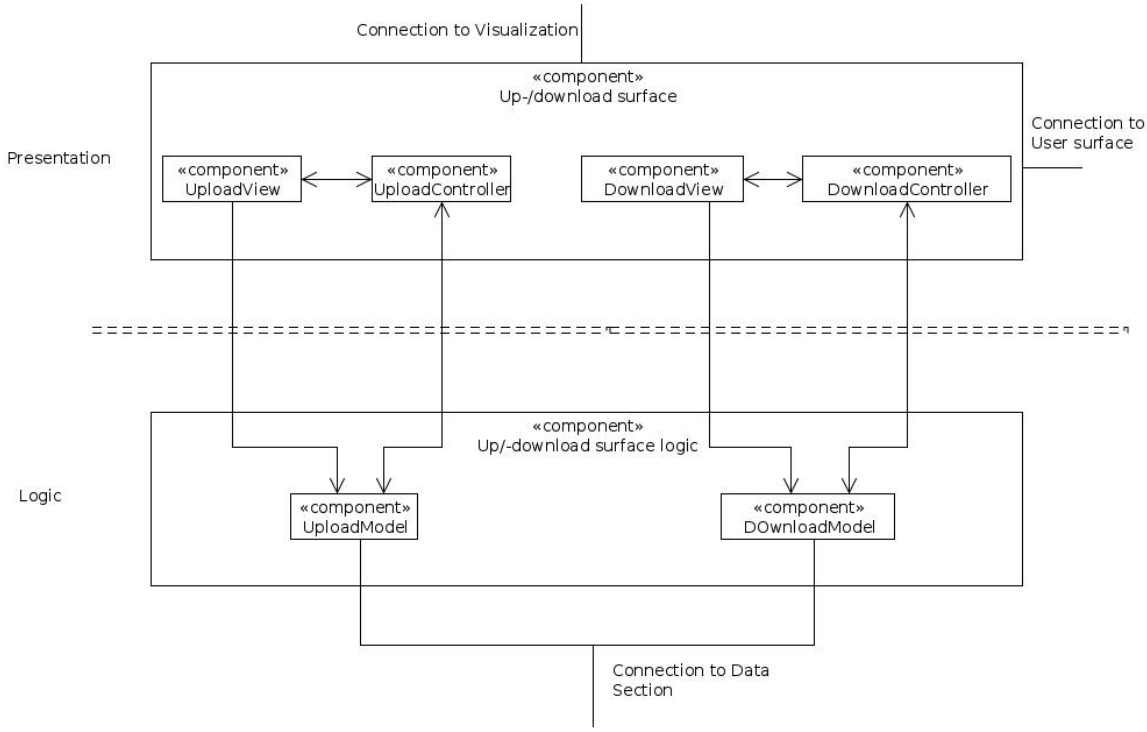

Figure 3.4: Up-/download surface zoom

representation.

3.2.1 Server

For the server we used the pattern layer again. The first level is the web-application-interface, which realizes the communication between server and web-application. The second level is the request-form level. Any kind of request to the server uses forms, for example the login uses a login form or to upload test data, we have an upload-data form. This level checks these forms for their correctness. Below that we have the processing level, which performs the given request. Lastly we have the database itself, which saves all given and calculated data. For adding new algorithms and general extensibility we used a combination of the microkernel and the reflection pattern. The microkernel pattern is the only pattern that allows adding new functionality to the system and guaranties the portability to other systems. The reflection pattern allows

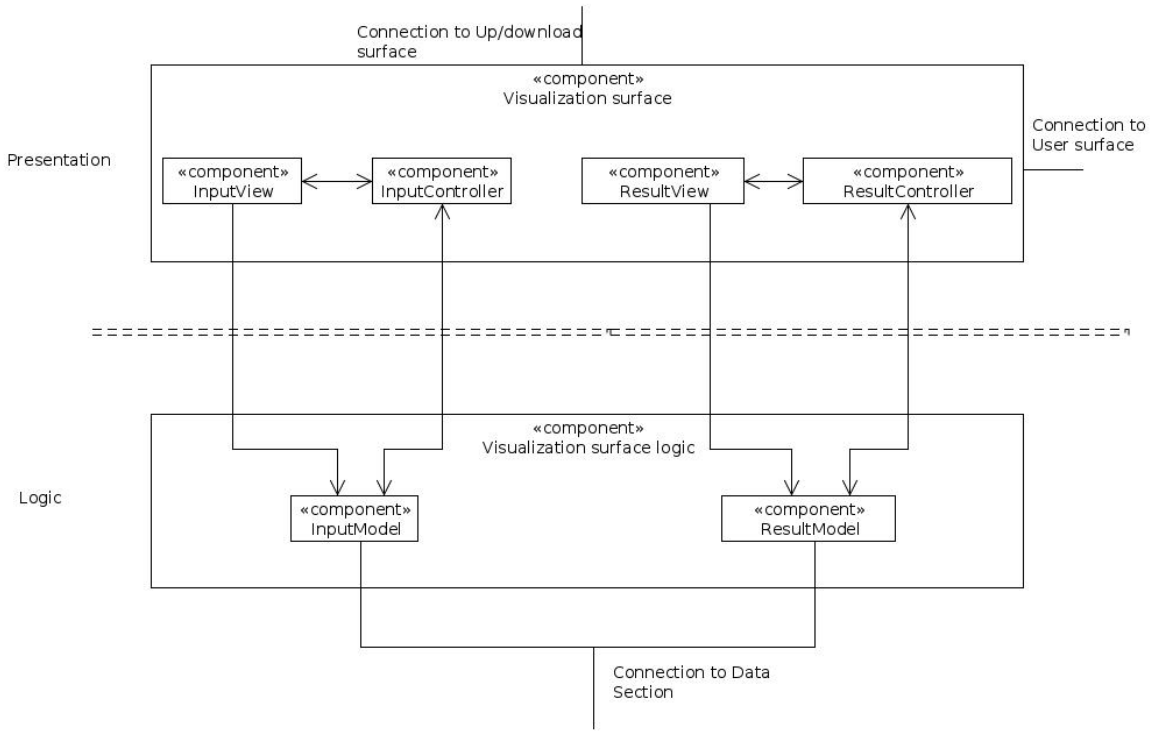

Figure 3.5: Visualization surface zoom

us to get the metadata of the algorithm. This way the algorithm just need to implement an interface which informs that what information the system needs. It does not need to know how the system later uses algorithm or how the system works in general. The other way around its is easy for the system to read what kind of parameters the algorithm needs to work properly (cf. Fig.3.7).

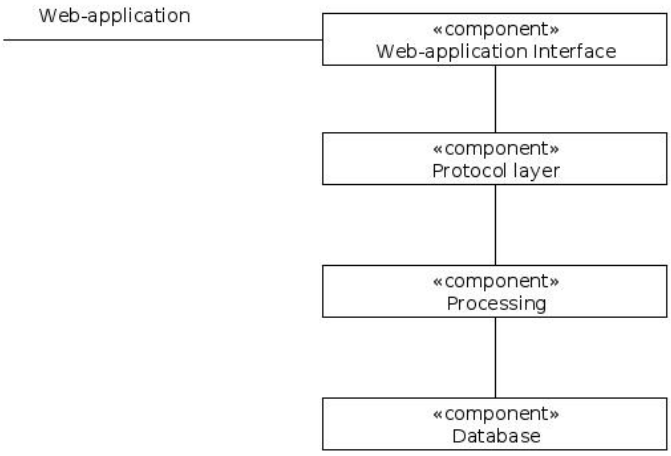

Figure 3.6: Server view

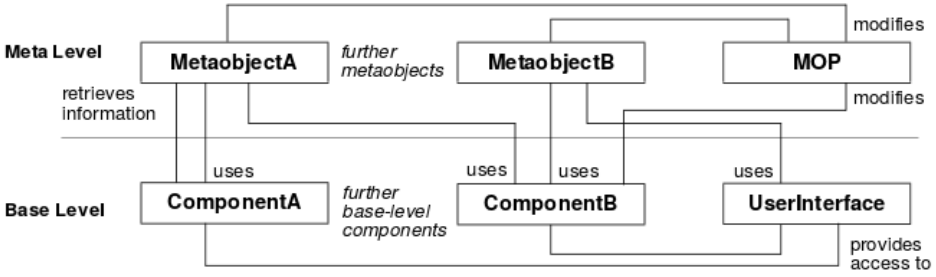

Figure 3.7: The reflection pattern based on http://wiki.ifs.hsr.ch/APF/files/Reflection.pdf

Chapter 4

Unused Patterns

Repository-Pattern: Not used because we only have the web-application and not mobile apps or desktop clients. Therefore the server can be adjusted towards the web-application.
Broker-Pattern: Our system runs only on one server and not multiple.

Chapter 5

Interaction Layer

The following section attends to the interaction between the different aleready mentioned system components in different scenarios. The main task is the visualization of the data exchange with sequence-diagrams; without an exact definition of the called methods.