

# Design Document for iExperiment

Peter Strauch

Kevin Jacob  
Jonas Barde

Johannes Kunz

December 2, 2015

# Contents

<b>I</b>	<b>Architectural Design</b>	<b>1</b>
<b>1</b>	<b>Einleitung</b>	<b>2</b>
<b>2</b>	<b>Externe Sicht</b>	<b>3</b>
<b>3</b>	<b>Struktursicht</b>	<b>4</b>
3.1	Allgemein . . . . .	4
3.2	Client . . . . .	5
3.3	User Area . . . . .	6
3.4	Analysis Perspective . . . . .	7
3.5	Server . . . . .	8
<b>4</b>	<b>Interaktionsschicht</b>	<b>10</b>
4.1	Allgemein . . . . .	10
4.2	Visualisierung eines Ergebnisses . . . . .	10

Part I

**Architectural Design**


# Chapter 1

## Einleitung

In diesem Dokument wird das zu entwickelnde System unter verschiedenen Anforderungen betrachtet und in Subsystem unterteilt. Dabei wird sich auf die Functional Requirements des Requirement Document vom 12.11.2015 bezogen.

## Chapter 2

# Externe Sicht

iExperiment verwendet als Bibliothek „Weka Workbench“ (FR002). Dabei wird die API der Bibliothek verwendet. Es ist nicht nötig, dass „Weka Workbench“ das System iExperiment kennt. Als externes System wird iExperiment selbst verwendet, allerdings auf anderen unabhängigen Servern. 

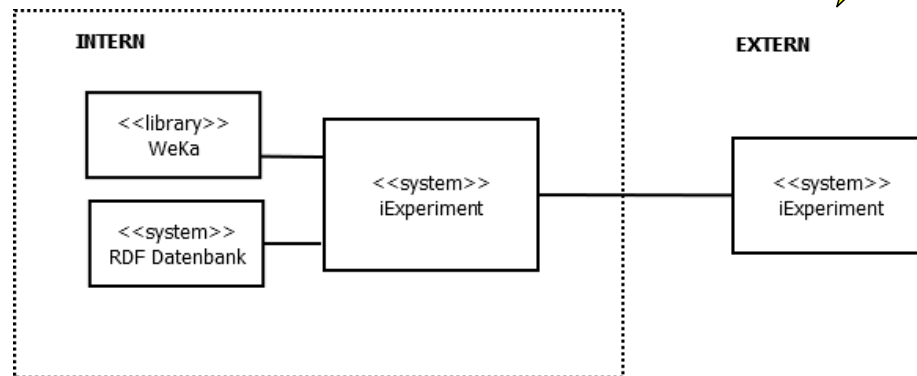


Figure 2.1: Kontextdiagramm

## Chapter 3

# Struktursicht

### 3.1 Allgemein

Nun wird die Architektur des Systems beschrieben, dafür muss die Struktur des Systems detaillierter beschrieben werden. Das System verwendet verschiedene Komponenten, die auf einer gemeinsamen Datenbasis basieren. Dies ist für die Systemintegrität notwendig. Unser System ist nur die Web-Applikation. Die Datenbasis besteht aus Messdaten aus Experimenten, Trainingsdaten, Modellen und Algorithmen.

Für diese Anforderungen ist das Repository Pattern hervorragend geeignet. Dabei wird die Datenbasis vom Repository gestellt und der Client operiert auf diesen Daten. Wenn die Laufzeit betrachtet wird, wird ersichtlich, dass es

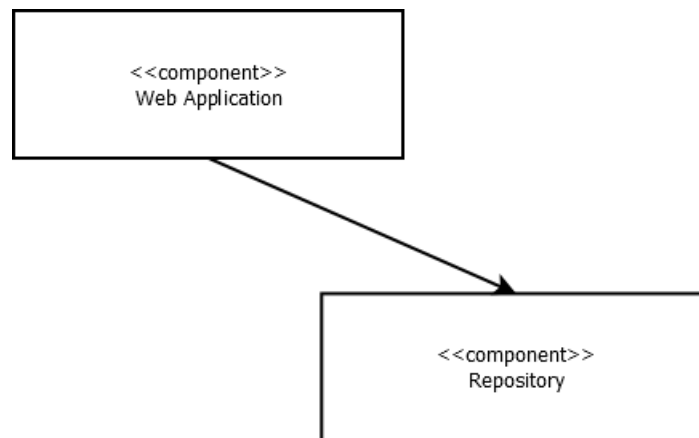



Figure 3.1: Repository Pattern

sich bei der Struktur zwischen den Komponenten Web-Applikation und Repository um ein Client-Server-Pattern handelt. Der Server bietet den Nutzern ver-

schiedene Funktionen an, wie Verarbeitung der Daten zu einem Modell, Vorhersagen und Visualisierung von Ergebnissen. Diese Funktionen sollten unabhängig vom Standort des Benutzers nutzbar sein.



Figure 3.2: Repository ist der Server

Außerdem bietet sich das Repository Pattern an, da Algorithmen hinzugefügt werden können. Dies muss nur auf dem Server geschehen, sodass garantiert ist, dass alle Clients die aktuellste Version, bzw. Zugang zu allen Algorithmen haben. 

## 3.2 Client

Das Layer Pattern wird verwendet um die Struktur des Dateisystems abstrakt darzustellen. Dabei kann der Client in drei Schichten eingeteilt werden:

- Präsentation
- Verarbeitungsmechanismen
- Datenschicht

Dabei ist die unterste Schicht für die Datenbank zuständig. Sie speichert und lädt Daten über das Internet. Die Präsentationsschicht besteht aus zwei Komponenten: Der User Area und der Analysis Perspective, die die analysierten Daten, bzw. Ergebnisse visualisiert.

Durch Verwendung des Layer Pattern können Änderungen an einzelnen Komponenten vorgenommen oder komplette Komponenten ausgetauscht werden, ohne dass andere Komponenten davon beeinflusst werden. Lediglich die Schnittstellen müssen unverändert bleiben, damit die darüber- oder darunterliegende Schichten unbeeinflusst bleiben.

Beim Client sind hauptsächlich die Präsentations- und Logikschicht interessant. Deshalb werden diese hier genauer beschrieben.

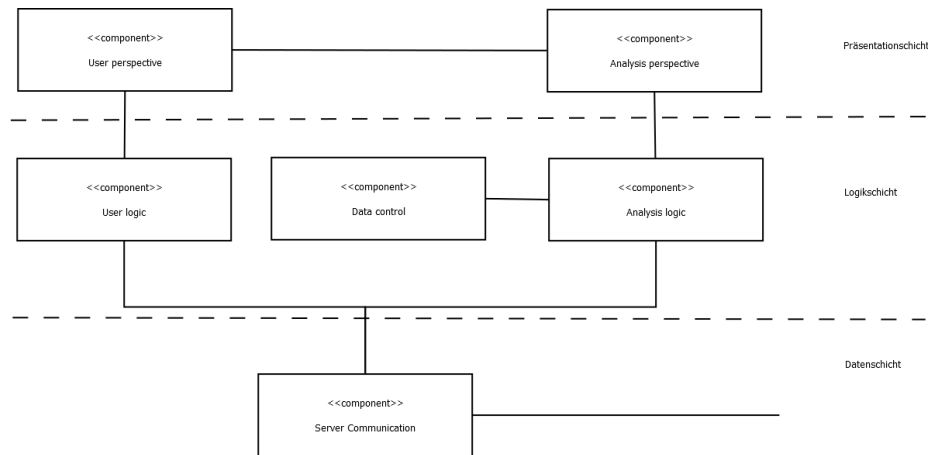


Figure 3.3: Struktur des Systems

### 3.3 User Area

In diesem Abschnitt wird die User Area Perspective und die darunterliegende Logik-Komponente erklärt. Die User Area Perspective wird in drei Ansichten eingeteilt. Dabei ist eine Ansicht für den Login/Registrierungs-Mechanismus zuständig. Die nächste Ansicht ist für den Nutzer, in der er verschiedene Verwaltungsfunktionen nutzen kann. Die dritte Ansicht ist für den Administrator und die dazugehörigen Funktionen, wie Nutzer- und Serververwaltung. In der User Area Perspective befindet sich „unter den drei Ansichten der Login controller, der die Daten der Login Ansicht verarbeitet. Diese Ansichten werden nach dem Model-View-Controller-Pattern strukturiert.

Das Modell dient der Umsetzung der Geschäftslogik, bspw. der des Login-Ablaufs für Nutzer oder Administratoren. Außerdem kann das Modell Daten aus der darunterliegenden Schicht anfordern, aber es können auch Daten gespeichert werden, die von der View Komponente dargestellt werden.

Die Präsentationsschicht nimmt Benutzeraktionen entgegen, ist allerdings nicht für deren Verarbeitung zuständig. Diese Schicht kennt ihren Controller und ihr Modell.

Der Controller verwaltet die Präsentation, d.h. er nimmt Benutzeraktionen entgegen, wertet diese aus und führt eine entsprechende Handlung aus. Die drei Ansichten haben einen gemeinsamen Controller, der dafür sorgt, dass Interaktionen mit dem Benutzer wirksam sind.



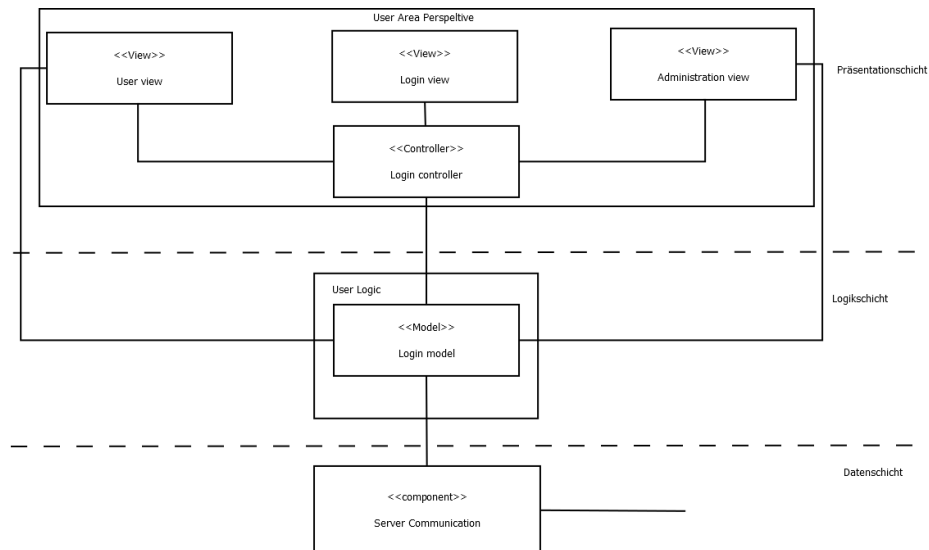


Figure 3.4: Darstellung der User Area Perspective

### 3.4 Analysis Perspective

Ein weiterer Teil der Client-Struktur ist die Analysis Perspective. Damit die User Area Perspective die Analysis Perspective starten kann, muss sie ein spezielles Interface der Analysis Perspective kennen. Solange dieses Interface gleich bleibt, kann sich die Analysis Perspective grundlegend ändern, ohne dass die User Area Perspective davon beeinflusst wird.

In den Modellen wird die Logik des Programmablaufs implementiert und die Daten gesichert. Input View und Input Controller sind dafür da, dass der Nutzer seine eigenen Datensätze, Algorithmen oder Modelle auswählen kann. Dort können auch verschiedene Parameter für die verschiedenen Algorithmen gesetzt werden. Die Output Komponente ist für die Ausgabe der Ergebnisse, bzw. der aus den Daten erstellten Modelle zuständig. Die Komponente Data control überprüft, ob die Daten dem richtigen Format entsprechen.

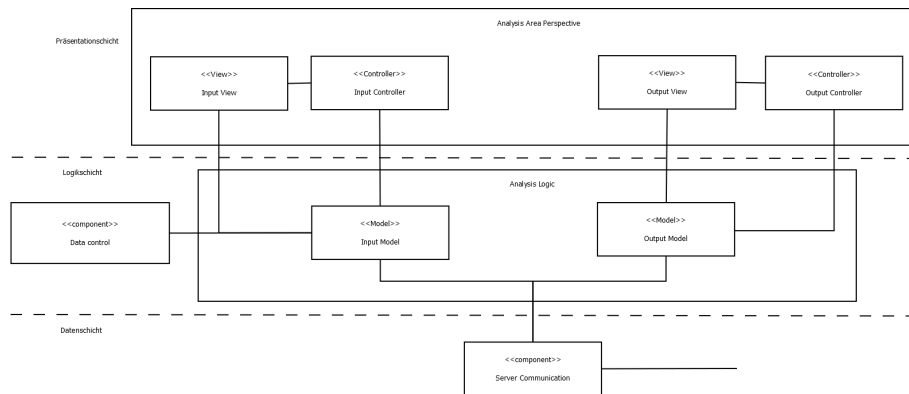


Figure 3.5: Darstellung der Analysis Perspective

### 3.5 Server

Der Server wird auch nach Vorbild des Layer pattern strukturiert. Der Administrator hat eine eigene Oberfläche in dem er Nutzer und Daten verwalten kann (FR028). Clients können von außen mit der Logikschicht über ein Interface kommunizieren. Mit Hilfe des Protocol Parser werden die Anfragen analysiert und die passende Logik Komponente aufgerufen.

Beim Login-/Registrierungsvorgang muss die User-Verwaltung mit der Datenbank kommunizieren können. Außerdem legt die User-Verwaltung die Rechte der Nutzer fest. Diese Rechte können dann von der User-Logik abgerufen werden. Soll aus einem Datensatz ein Modell oder ein Ergebnis erstellt werden, ist die Analysis Logic nötig.

Dabei ist wichtig, dass die Komponenten nicht direkt mit der Datenbank kommunizieren, sondern mit einer Abstraktionskomponente, sodass eine Komponente die Datenbanktechnologie nicht kennen muss, um auf die Datenbank zuzugreifen.

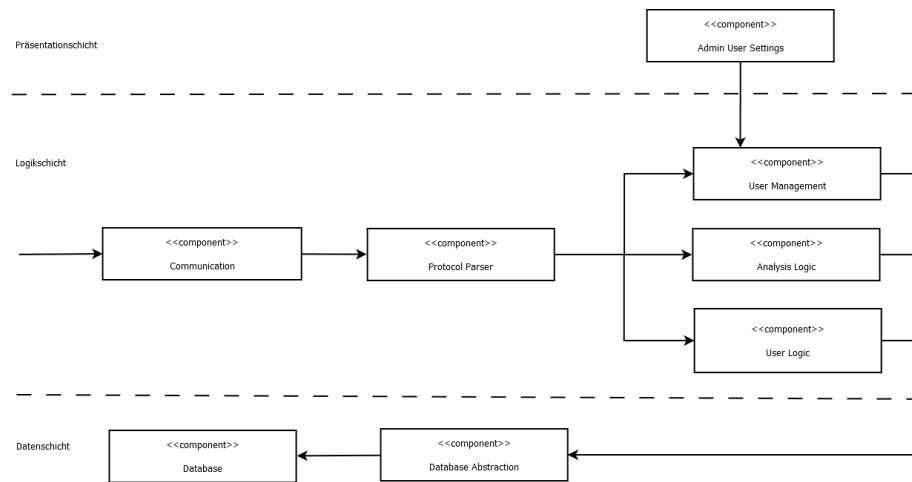


Figure 3.6: Struktur des Servers

## Chapter 4

# Interaktionsschicht

### 4.1 Allgemein

In diesem Kapitel werden verschiedene Szenarien detaillierter beschrieben. Dabei ist zu beachten, dass es sich nicht um Methodenaufrufe handelt, sondern nur Vorgänge beschreibt, die die Struktur veranschaulichen sollen.

### 4.2 Visualisierung eines Ergebnisses

Hier wird schematisch gezeigt, wie aus eingegeben Daten ein Ergebnis erstellt wird, z.B. in Form eines Diagramms, Baums oder ähnlichem.

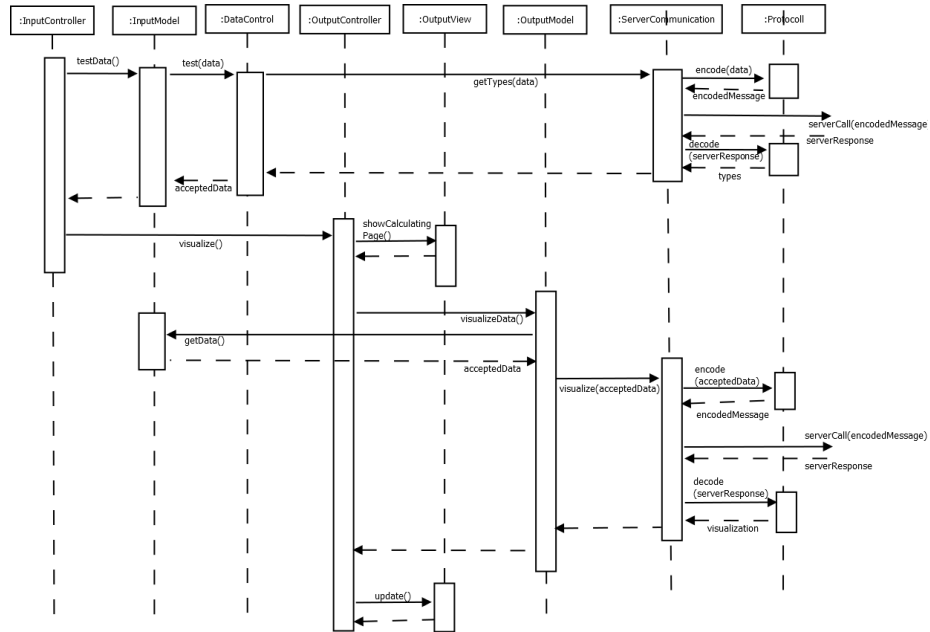


Figure 4.1: aus Sicht des Clients

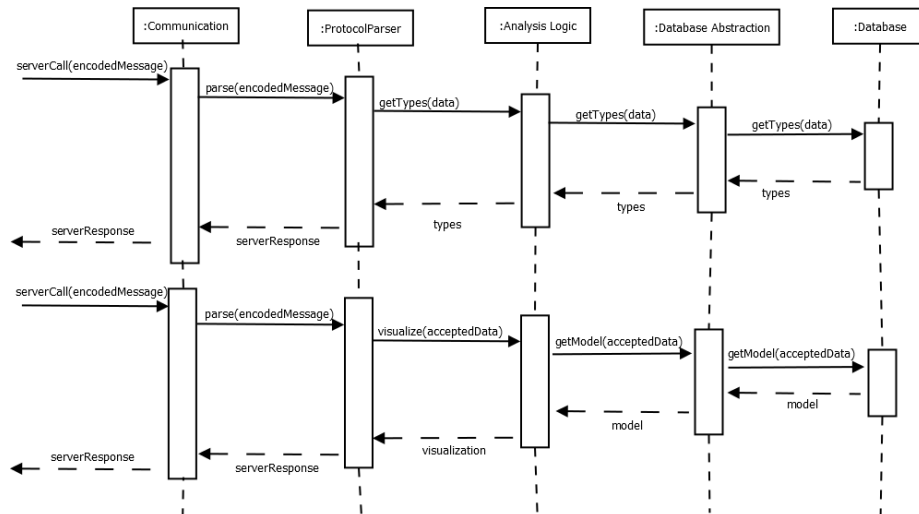


Figure 4.2: aus Sicht des Servers