Architekturdokument

AtmoCalc

Nicole Hauck Hanna Heinemann Andreas Luft Lars Porth

SE 2014/2015

Einleitung

In diesem Architekturdokument wird das zu entwickelnde System „AtmoCalc“ unter verschiedenen Perspektiven betrachtet und in einzelne Teile zu zerlegt. Alle Referenzen auf das Anforderungsdokument beziehen sich dabei auf das bereitgestellte Dokument vom 09.12.2014. Die Referenzen beziehen sich auf die Funktionalen Anforderungen des Dokuments.

Externe Sicht

Das zu entwickelnde System muss in Abgrenzung anderer Systeme betrachtet werden. AtmoCalc nutzt eine Reihe weiterer Systeme, um dem Nutzer spezielle Informationen anzeigen zu können (betrifft FR065). Es wird also die API der externen Systeme benutzt. Die externen Systeme müssen dazu die AtmoCalc-Anwendung nicht kennen (siehe Abbildung 1).

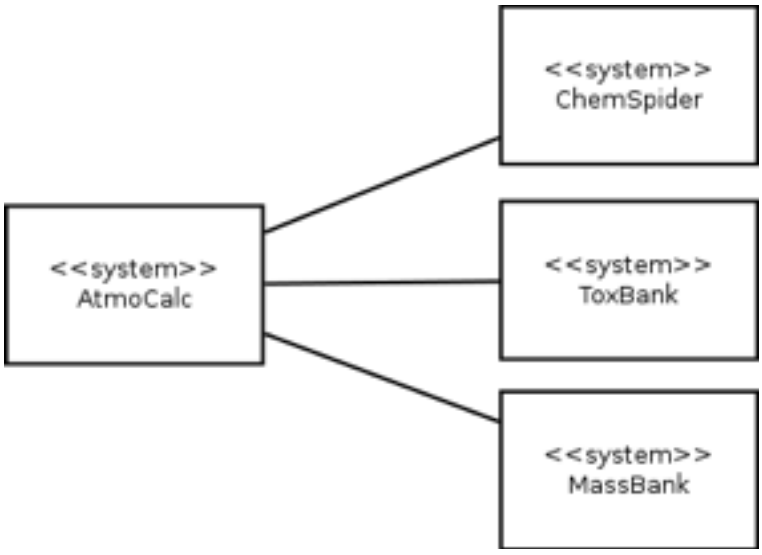

Abbildung 1: Context-Diagramm im Bezug auf andere Systeme

Struktursicht

Allgemein

Um das System weiter zu beschreiben, muss die Struktur des Systems genauer beschrieben werden. Auf der höchsten Granularitätsebene existieren dabei verschiedene Komponenten im System, die auf einer gemeinsamen Datenbasis operieren müssen. Als verschiedene Komponenten können in diesem Fall die Web-Applikation oder ein extern entwickelter Client gesehen werden. Die gemeinsame Datenbasis sind die Informationen über Regeln und Strukturen. Die Komponenten sind unabhängig voneinander, sie wissen also nichts über die Existenz der anderen Komponenten. Im System können große Mengen von Daten vorhanden sein, die wahrscheinlich über einen langen Zeitraum gespeichert werden müssen.
Diese Anforderungen können vom Repository-Pattern sehr gut abgedeckt werden. Das Repository stellt dabei die Datenbasis während die Clients auf diesen Daten operieren (siehe Abbildung 2).

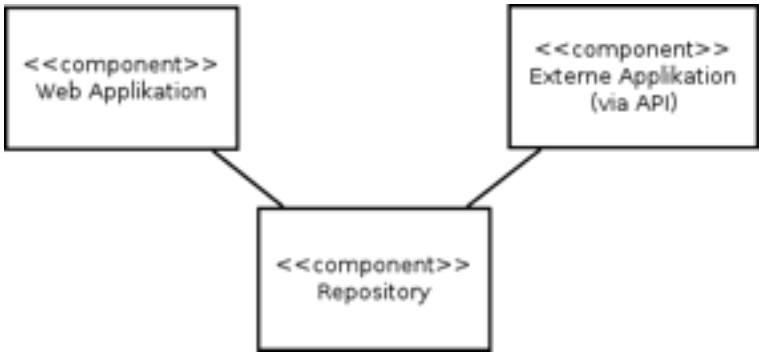

Abbildung 2: Auf der obersten Granularitätsebene lässt sich das System mit dem Repository-Pattern darstellen

Betrachtet man die Struktur des Systems zur Laufzeit, so wird klar dass es sich, aus der Sicht einer einzelnen Komponente, um ein Client-Server-Pattern handelt. Das Repository bietet den Clients verschiedene Services an, beispielsweise die Möglichkeit einen Datensatz zu analysieren, welche die Clients nutzen. Auf die Daten muss prinzipiell von überall auf der Welt zugegriffen werden können. Außerdem soll das entwickeln externer

Anwendungen, die das AtmoCalc-System über eine API nutzen, ermöglicht werden (vgl. FR001). Aus diesem Grund wird auch das Client-Server Pattern genutzt, wobei das Repository den Server darstellt (siehe Abbildung 3).

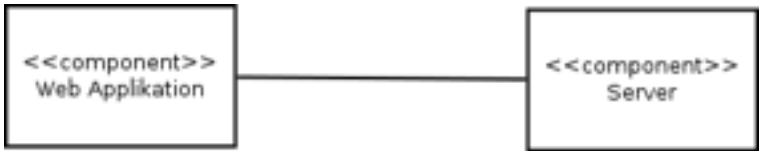

Abbildung 3: In diesem Fall ist die Web-Applikation der Client

Client

Eines der häufigsten Pattern, das genutzt wird um ein Softwareprojekt zu strukturieren ist das Layer-Pattern. Der Client kann in 3 Schichten eingeteilt werden. Die oberste Schicht dient der Präsentation, die mittlere der Programmlogik und die unterste ist die Datenschicht. Da laut NRF021 die Berechnungen auf dem Server stattfinden sollen, besitzt diese lediglich eine Komponente zur Kommunikation mit dem Server. Da die Kommunikation in definierten Formaten ablaufen soll, beispielsweise PDF beim Export (vgl. FR061), wird eine weitere Komponente eingefügt, die das Verpacken der Nachrichten bzw. Entpacken der Antworten übernimmt. Die Logikschicht ist relativ dünn. Ihre Hauptaufgabe besteht darin, den Ablauf des Programms zu steuern. Die Präsentationsschicht besteht aus Komponenten für 3 geforderte Perspektiven. Zum einen die User Area (vgl. FR010 bis FR023) sowie die Analysis Perspektive (vgl. FR039 bis FR051), in der beispielsweise die Art (siehe Abbildung 4).

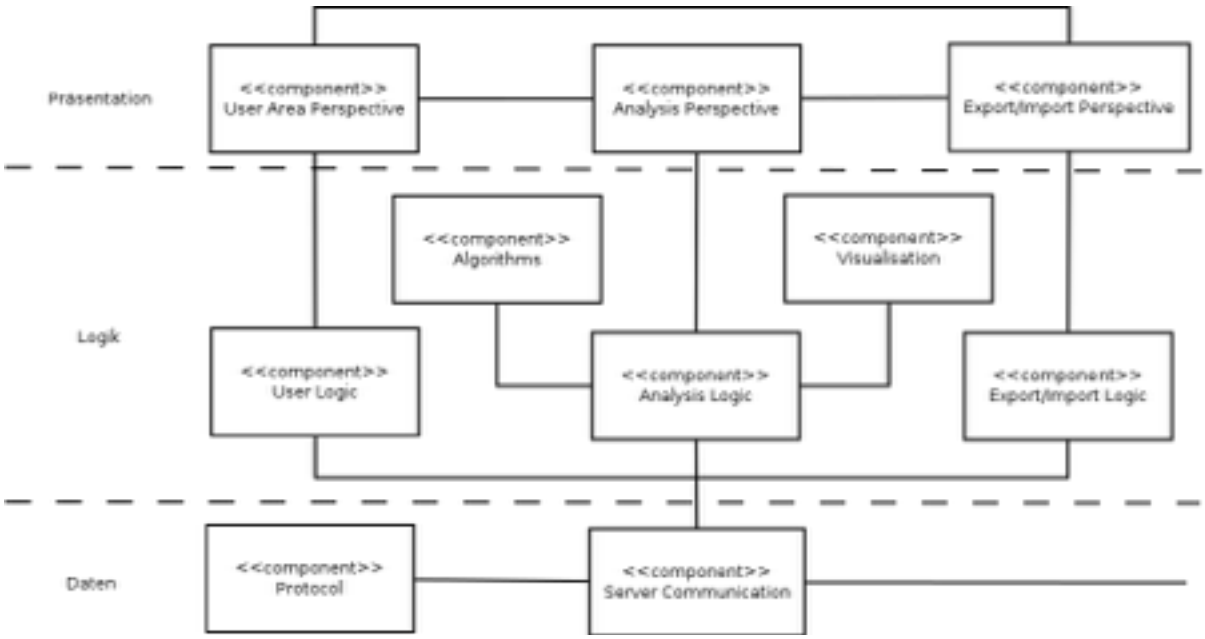

Abbildung 4: Grobübersicht über den Client

Der Vorteil dieses Patterns ist, dass sich einzelne Layer komplett austauschen lassen, solange die Interfaces gewahrt bleiben. Da laut dem Layer-Pattern eine Kommunikation nur nach unten erlaubt ist und sich die Logik dabei nicht ändert, bleiben die Interfaces stabil. Da sich die Daten nur auf Anfrage ändern, also nach einem Aufruf in der Präsentationsschicht, funktioniert dies auch problemlos. Die Assoziationen zwischen den Komponenten der Präsentationsschicht kommt dienen dem Wechsel der Perspektive.

Von Interesse sind beim Client hauptsächlich die Präsentations- und Logikschicht. Aus diesen Gründen wird diese hier weiter verfeinert.

Zunächst wird die „User Area"-Perspektive und die dazugehörigen Logik Komponente betrachtet. Die „User Area"-Perspektive kann in zwei Sichten aufgeteilt werden: Eine Sicht wird für den Login-/Registrierungs-Mechanismus benötigt, eine weitere für die Verwaltungsfunktionen, die jedem Nutzer bereitgestellt werden (FR022 – FR029). Beide Sichten lassen sich mithilfe des MVC-Patterns realisieren. Das Model dient in diesen Fällen hauptsächlich zur Implementierung der Logik (der Business Logic), beispielsweise im Login-Ablauf. Aber es fragt auch nach Daten bei der Datenschicht bzw. speichert Daten, die die View-Komponente dann präsentiert (siehe Abbildung 5).

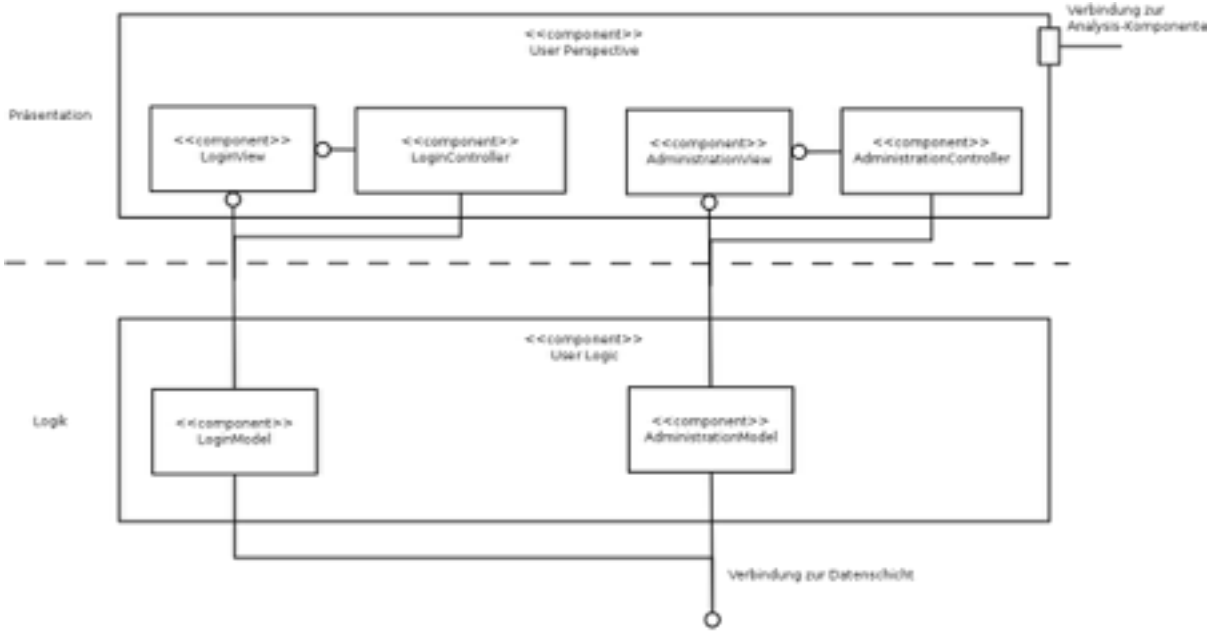

Abbildung 5: Ein detaillierter Blick auf die User-Komponente

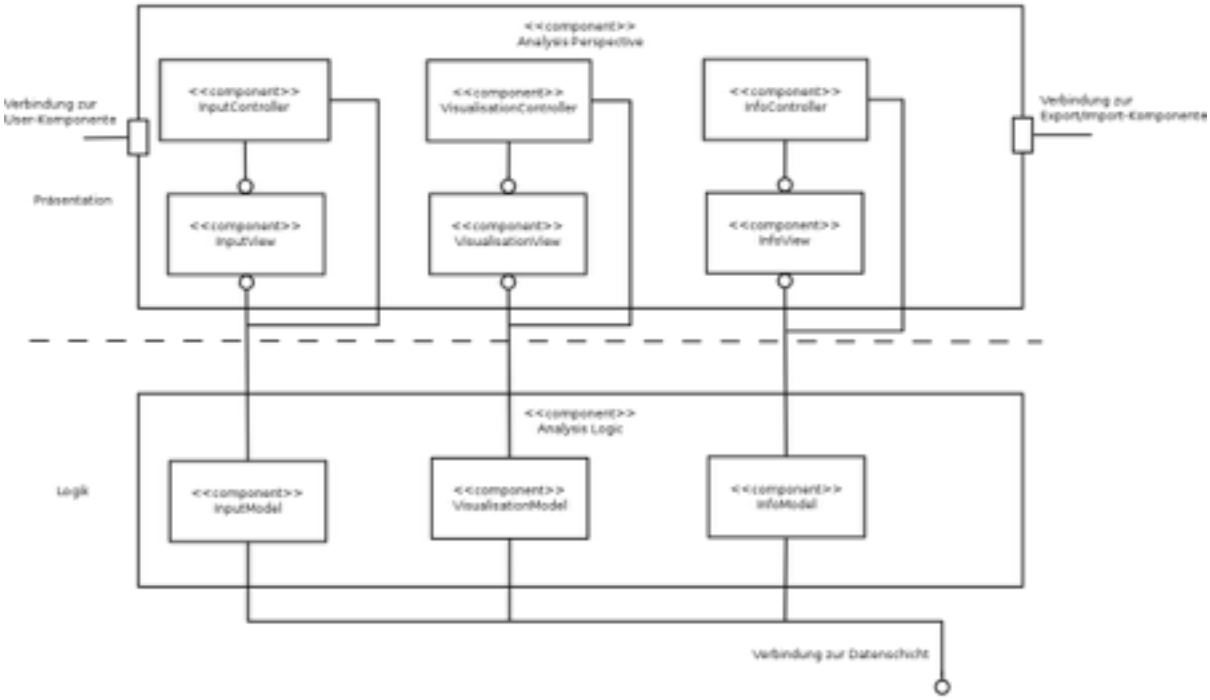

Abbildung 6: Ein detaillierter Blick auf die Analysis-Komponente

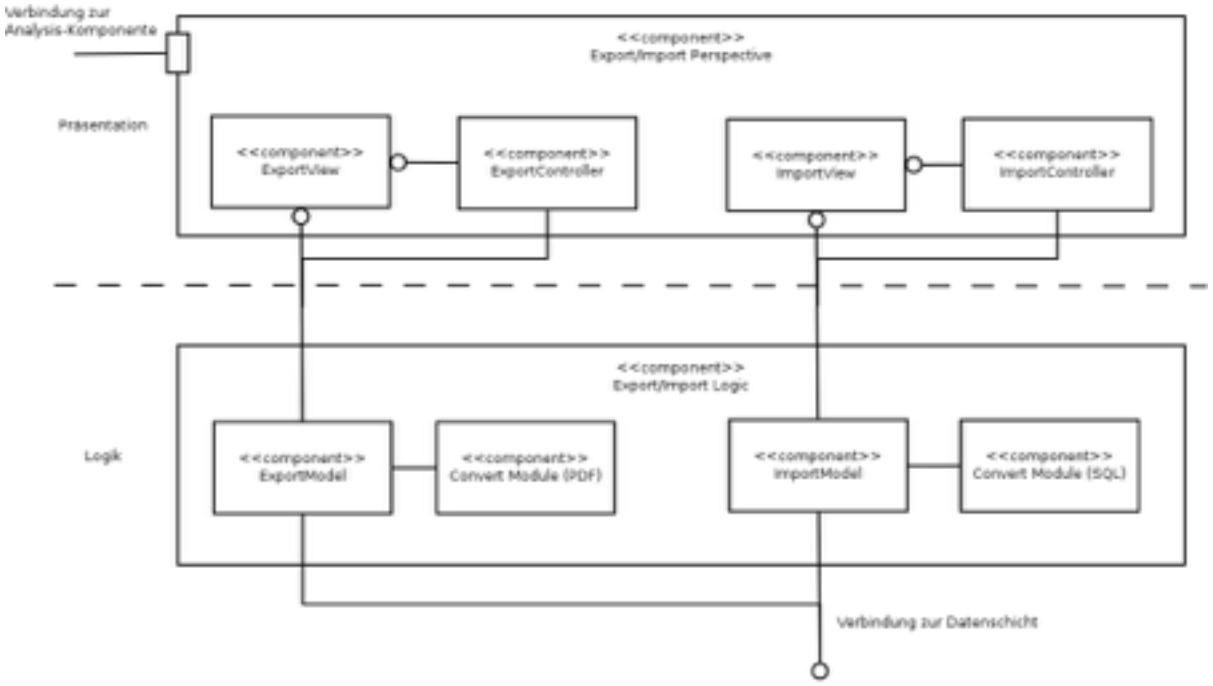

Abbildung 7: Ein detaillierter Blick auf die Export/Import-Komponente

Server

Auch der Server ist nach einem Schichtenmodell aufgebaut. Nach FR031 stellt der Server eine GUI für die Administration bereit. Laut FR036 und FR037 können über diese Oberflächen User und Datensätze verwaltet werden. Die Logikschicht bietet das Interface nach außen, über das die Clients mit dem Server kommunizieren. Einkommende Nachrichten werden vom Protocol-Parser analysiert, welcher danach die entsprechende Logik-Komponente aufruft. Diese können entweder das User-Management sein, die Logik hinter der Analyse eines Datensatzes, die Logik hinter dem Abfragen weiterer Informationen in den externen Datenbanken oder der Datensatzverwaltung.
Die User-Verwaltung kann mit der Datensatzverwaltung kommunizieren, damit ein Nutzer einstellen kann welche Datensätze er analysieren möchte. Beim Login oder der Registrierung muss die User-Verwaltung auch mit der Datenbank kommunizieren. Um einen Datensatz zu analysieren wird die Analyse-Logik Komponente genutzt. Diese kann mit der Datensatzverwaltung und der Algorithmenverwaltung kommunizieren um die Datensätze und die Algorithmen, die die Vorschriften für die Analyse enthalten, für einen bestimmten Nutzer anzufordern. Außerdem muss sie mit der Visualisierungs-verwaltung kommunizieren, um Analysen visualisieren zu können. Die Datensatzverwaltung und die Algorithmenverwaltung kann mit der Datenbank kommunizieren, um Datensätze, Regeln und Algorithmen aus der Datenbank zu laden.
Die Komponenten kommunizieren dabei nicht direkt mit der Datenbank, sondern mit einer AtmoCalc-spezifischen Abstraktionskomponente, die dafür sorgt dass die anderen Komponenten die dahinter liegende Datenbanktechnologie nicht kennen müssen.
Für die Kommunikation mit den externen Datenbanken wird eine weitere Komponente in der Datenschicht benutzt. Diese bündelt die API's der externen Datenbanken und gibt deren Informationen an die Logikkomponente zurück. Die Logikkomponente für die Einbindung externer Informationen organisiert die Aufrufe (siehe Abbildung 9).