HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

INSTITUT
FÜR
SOFTWARE

Web UI-Front-End for Fluid Dynamics Cloud

Bachelorarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2011

Autoren:	Anita Hollenstein
	Patrice Müller
Betreuer:	Prof. Dr. Luc Bläser
Projektpartner:	Dr. Djamel Lakehal, Ascomp GmbH

Impressum

Kontakt
Anita Hollenstein
Patrice Müller

Druckdatum: 23.12.2011

Erklärung über eigenständige Arbeit

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben.

Rapperswil, den 23.12.2011

Anita Hollenstein **Patrice Müller**

Abstract

Ausgangslage
Die ASCOMP GmbH ist mit dem Fluiddynamik Berechnungsprogramm TransAT international tätig. Das Ziel von ASCOMP ist es, TransAT in eine Rechner-Cloud auszulagern und über eine webbasierte grafische Benutzeroberfläche auf einfache Weise zugänglich zu machen. Die Aufgabe dieser Bachelorarbeit ist es, eine Machbarkeitsstudie über diese webbasierte grafische Benutzeroberfläche durchzuführen.

Vorgehen / Technologien
Die Anforderungen des neuen WebUI Front-Ends, also der Benutzeroberfläche, sind zuerst analysiert worden. Darauf basierend wurde ein Prototyp erstellt, welcher sich auf die wesentlichen Konzepte und die technischen Risikopunkte des WebUI Front-Ends konzentriert. Der Prototyp wurde an eine Cloud angebunden, auf welcher das TransAT Programm läuft. Die Benutzeroberfläche ist eine Rich Internet Applikation, die mit Microsoft Silverlight direkt in verschiedenen Browsern läuft. Für die Cloud-Anbindung wurde zuerst beabsichtigt, den Windows basierten High Performance Cluster (HPC) der HSR zu benutzen. Der HPC war erst seit kurzer Zeit in Betrieb und noch nicht darauf ausgerichtet, von ausserhalb des HSR-Netzwerks benutzt zu werden. Zur Überbrückung dieser Einschränkung musste daher zunächst ein Zwischen-Service entwickelt werden. Jedoch ist TransAT wider Erwarten nicht rechtzeitig von anderen Entwicklern auf den HPC portiert worden. So wurde als Ersatz ein Mockup des HPC's geschrieben, welcher die Berechnung des TransAT Programms simulierte. In der zweiten Hälfte der Bachelorarbeit gab Cloudbroker, der zweite Cloud-Partner dieser Arbeit, die Fertigstellung der Schnittstelle zu seiner Cloud-Infrastruktur bekannt. Schliesslich ist es gelungen, Cloudbroker funktionsfähig an den Prototypen anzubinden und für die TransAT-Berechnung der Simulationsprojekte aus dem WebUI Front-End einzusetzen.

Ergebnis
Das Ergebnis der Arbeit ist ein Prototyp, welcher die folgenden Funktionalitäten unterstützt und so die Machbarkeit des Web-Front-Ends für Fluiddynamik-Simulationen in der Cloud demonstriert:

- Übersicht und Verwaltung der Simulationsprojekte
- Reduzierte Parameter-Editierung mit grafischer und funktionaler Unterstützung des Arbeitsflusses
- Upload von Projekten auf eine Cloudbroker-Cloud zur Berechnung
- Download der berechneten Lösung mit der Möglichkeit, diese lokal darzustellen
- Schnittstelle zur Cloud-Infrastruktur von Cloudbroker
- Schnittstelle zum HPC der HSR mit simuliertem HPC Berechnungsablauf

Inhaltsverzeichnis

TEIL I: TECHNISCHER BERICHT

Teil I

Technischer Bericht

1 Einführung

1.1 Aufgabenstellung

Die folgenden Unterkapitel der Einführung wurden aus der Aufgabenstellung der Bachelorarbeit entnommen. Somit ist der nachfolgende Teil dieses Abschnitts (1.1) als Zitat von Herrn Prof. Dr. Luc Bläser, dem Betreuer dieser Bachelorarbeit, gekennzeichnet.

1.1.1 Ausgangslage

Die Ascomp GmbH ist ein ETH Physik-Spin-Off, welche im Gebiet der Complex Fluid Dynamics Modellierung und Simulation tätig ist. Um komplexe Fluid Dynamics Simulationen zu rechnen, bedarf es der Ausführung in einem High-Performance-Computing Cluster. Ein solcher Cluster steht insbesondere an der HSR zur Verfügung und könnte zum Beispiel auch im Sinne des Cloud Computings für Fluid Dynamics Simulationen der Firma Ascomp genutzt werden.

Das Durchführen der Simulationen erfordert im Wesentlichen drei Schritte: die Eingabe des Modells mit vielzähligen Parametern (in der Grössenordnung von 1000), das Ausführen der Simulation auf dem HPC-Cluster sowie die Darstellung und Analyse der gerechneten Simulationsresultate. Dieser Prozess findet in der Regel iterativ statt, d. h. die Parameter können jeweils wieder angepasst und die Simulation erneut laufen gelassen werden.

Die Firma Ascomp hat bereits eine UI-Anwendung entwickelt, welche als lokales Programm läuft und die Spezifikation der Fluid Dynamics Simulationen ermöglicht. Die existierende Anwendung unterstützt allerdings keine Cloud-Ausführung, hat noch ungenügende Usability und ist auch in einer etwas älteren Technologie (FLTK) entwickelt.

Die Firma Ascomp ist darauf bestrebt, zukünftig eine komfortable und moderne Benutzerschnittstelle für die Unterstützung des gesamten Simulationsprozesses (Eingabe, Ausführung, Ausgabe) anzubieten. Hierzu sollte ein Web-Interface zur Verfügung gestellt werden, so dass es im Browser von Clienten-Rechner ohne Installation einer lokalen Anwendung flexibel benutzt werden kann.

Das Ziel dieser Bachelorarbeit ist es, eine Machbarkeitsstudie für das „Web UI-Front-End for Fluid Dynamics Cloud“ auf der Basis der .NET Silverlight Technologie durchzuführen. Dabei sind insbesondere folgende Aspekte relevant (nicht abschliessend):

1) Unterstützung des Simulations-Workflows mit dem UI-Front-End

2) Datenimport (verschiedene Formate) und Upload via UI

3) Parametereingabe (mit Enabling, Validierung, Konsistenzregeln, Kontextinformationen)

4) Darstellung von 2D und 3D Modellen

5) Schnittstelle zur Cloud (HPC Cluster der HSR, eventuell auch weitere wie „Cloudbroker“)

6) Ausführung auf dem Rechencluster

7) Überwachung und Abbruch einer laufenden Clusterrechnung

8) Rückgabe der Resultate und Download via UI

9) Visualisierung der Resultate

10) Flexible Anpassbarkeit des UI an neue oder geänderte Parameter

11) Unterstützung von verschiedenen Versionen der Simulations-Engine

12) Sicherheit bei der Übertragung (wenn nötig)

13) Optional: DB-Verwaltung der Eingaben und Resultate, z.B. mit History

In der Studie ist zuerst eine Grobabklärung aller wesentlichen Funktionalitäten, Schritte und Aspekte für den Gesamt-Workflow nötig und Grobkonzept auszuarbeiten. Die identifizierten Themen sind dabei nach Relevanz und Risiko zu priorisieren. Danach sollen die Themen nach Priorität vertieft analysiert werden und Lösungskonzepte dafür erarbeitet werden.

Aufgrund von Zeitgründen wird nach der Grobanalyse - unter Rücksprache mit dem Betreuer - entschieden, wie umfassend der gesamte Workflow erarbeitet wird oder ob der Fokus hauptsächlich auf die Eingabe gesetzt wird.

Die Machbarkeitsstudie soll als „Proof of Concept" einen exemplarischen Software-Prototyp für das Web UI-Front-End auf der Basis der .NET Silverlight Technologie beinhalten, der die Konzepte der Studie demonstriert. Der Prototyp umfasst zwar die essentiellen Schritte und Funktionalitäten, jedoch mit limitiertem Parameter- und Funktionalitätsumfang. Der Prototyp zeigt insbesondere Lösungen in den Bereichen auf, wo technische Herausforderungen und Risikopunkte identifiziert wurden.

(Prof. Dr. Luc Bläser: Aufgabenstellung der Bachelorarbeit, 19.09.2011)

1.1.2 Ziele und Aufgabenstellung

Die Aufgabe dieser Arbeit ist es, die Machbarkeitsstudie für das „Web UI-Front-End for Fluid Dynamics Cloud" für die Firma Ascomp durchzuführen.

Folgende spezifische Ziele werden vorgegeben:

- *Durchführung der Machbarkeitsstudie für das graphische Web-Front-End zur Fluid Dynamics Simulation in der Cloud. Dies umfasst:*
  - *Erste Phase: Grobanalyse und Grobkonzept für den gesamten Workflow mit den einzelnen erforderlichen und wünschenswerten Funktionalitäten und Eigenschaften*
  - *Zweite Phase: Vertiefte Analyse von priorisierten Themen und Erarbeitung von detaillierten Lösungskonzepten (z. B. Parametereingabe). Die Priorisierung ist mit dem Betreuer abzusprechen.*
  - *Ein Konzept für das UI-Design sowie eine SW-Architektur auf Basis von .NET Silverlight 4 und WPF*
  - *Liste der technischen Risikopunkte mit evaluierten Lösungsmöglichkeiten*
  - *Eine Aufwandschätzung für die vollständige Realisierung*

Das Front-End ist auf Basis von .NET Silverlight 4 mit WPF auszulegen und für das Cloud Computing soll primär der Microsoft HPC Cluster der HSR berücksichtigt werden.

- *Umfassende Dokumentation der Studie mit allen Analysen, Konzepten und Resultaten.*

- *Entwicklung eines Software-Prototyps mit beschränktem Umfang, der den Workflow, die wesentlichen Konzepte und die technischen Risikopunkte illustriert. Der Prototyp ist in .NET Silverlight 4 mit WPF zu implementieren. Es können geeignete existierende Bibliotheken oder Lösungen eingesetzt werden.*

(Prof. Dr. Luc Bläser: Aufgabenstellung der Bachelorarbeit, 19.09.2011)

1.1.3 Durchführung

Mit dem HSR-Betreuer und dem Auftraggeber finden in der Regel wöchentliche Besprechungen statt. Zusätzliche Besprechungen sind nach Bedarf durch die Studierenden zu veranlassen. Als technische Kontaktperson der Firma Ascomp steht Herr Daniel Caviezel zur Verfügung.

Alle Besprechungen sind von den Studenten mit einer Traktandenliste vorzubereiten und die Ergebnisse in einem Protokoll zu dokumentieren, das dem Betreuer und dem Auftraggeber per E-Mail zugestellt wird.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. An Meilensteinen gemäss Projektplan sind einzelne Arbeitsresultate in vorläufigen Versionen abzugeben. Über die abgegebenen Arbeitsresultate erhalten die Studierenden ein vorläufiges Feedback. Eine definitive Beurteilung erfolgt auf Grund der am Abgabetermin abgelieferten Dokumentation.

(Prof. Dr. Luc Bläser: Aufgabenstellung der Bachelorarbeit, 19.09.2011)

1.1.4 Dokumentation

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen. Die zu erstellenden Dokumente sind im Projektplan festzuhalten. Alle Dokumente sind nachzuführen, d.h. sie sollten den Stand der Arbeit bei der Abgabe in konsistenter Form dokumentieren. Die Dokumentation ist vollständig auf CD/DVD in 3 Exemplaren abzugeben. Auf Wunsch ist für den Auftraggeber eine gedruckte Version zu erstellen.

(Prof. Dr. Luc Bläser: Aufgabenstellung der Bachelorarbeit, 19.09.2011)

1.2 Umfeld der Arbeit

1.2.1 Rahmenbedingungen

1.2.1.1 Aufwand

Mit dieser Bachelorarbeit erwerben die Studenten Patrice Müller und Anita Hollenstein jeweils 12 ECTS-Punkte. Aus der allgemeinen Definition der ECTS-Punkte ergibt sich folgender Aufwand: 12 ECTS-Punkte à 30 Stunden= 360 Stunden pro Student über 14 Wochen. Somit entspricht dies einem Arbeitsaufwand von 25.7 Stunden pro Woche und Student.

1.2.1.2 Betreuung

Von der Seite der HSR wird die Arbeit von Herrn Prof. Dr. Luc Bläser des Instituts für Software betreut.

1.2.1.3 Ansprechpartner des Auftraggebers

Herr Daniel Caviezel ist Ansprechpartner des Auftraggebers Ascomp.

1.2.1.4 Wichtige Termine

In der Tabelle 1.1 sind grundlegende Termine des Starts und der Abgabe der Bachelorarbeit aufgeführt. Weitere, auf den Projektinhalt bezogene, Termine wurden in Form von Meilensteinen festgelegt und sind im Teil 3: Projektmanagement dieses Berichts aufgeführt.

Datum	Termin
19. September 2011	Start der Bachelorarbeit
23. Dezember 2011	Abgabe der Bachelorarbeit

Tabelle 1.1: Wichtige Termine

1.2.2 Infrastruktur

1.2.2.1 Räumlichkeiten

Grundsätzlich wird der reservierte Raum 2.006 für die Informatik Bachelorarbeit benutzt. Zusätzlich stellt die Firma Ascomp ihre Räumlichkeiten zur Verfügung, um gemeinsame Arbeiten zu erleichtern. Weitere Arbeit kann auch zuhause erfolgen.

1.2.2.2 Hardware

Gerät	Beschreibung
1x T410	Privater Laptop von Patrice Müller
1x XPS M1530	Privater Laptop von Anita Hollenstein
2x CELSIUS W380	Persönliche Rechner im Arbeitsraum (von der HSR zur Verfügung gestellt)
2x Server	Ubuntu-Server 10.04, Kernel 2.6.32 (von der HSR zur Verfügung gestellt) Windows Server 2008 R2 (von der HSR zur Verfügung gestellt)
Drucker	Privat und von der HSR

Tabelle 1.2: Hardware

1.2.2.3 Software

Bereich	Komponente
Betriebssystem	• Microsoft Windows 7 Enterprise / Professional SP1 • Windows Server 2008 R2 • Ubuntu 11.04, Kernel 2.6.38 unter VMware Workstation 7.1.4
Entwicklungsumgebung und Tools	• Microsoft Visual Studio 2010 SP1 • Microsoft Expression Blend 4 • Silverlight Toolkit • Silverlight 4 SDK • Git Extensions • MVVM Light • StyleCop
Versionsverwaltung	• Git (Server der HSR)
Programmiersprachen	• C# • XAML • C++
Dokumentation und Tools	• Microsoft Word 2010

	• Microsoft Excel 2010 • Microsoft Visio 2010 • Inkscape 0.48.2 • Enterprise Architect • Adobe Photoshop CS5 • Dropbox 1.1.45
Projektmanagement	• Redmine 1.2.1 (Server der HSR)

Tabelle 1.3: Software

1.2.2.4 Organisation

Tool	Beschreibung
Redmine 1.2.1 (Server der HSR)	Ein Web-basiertes Projektmanagement-Tool

Tabelle 1.4: Organisation

1.2.3 Abgrenzungen

Bei dieser Arbeit handelt es sich um eine Machbarkeitsstudie. Aus diesem Grund wird kein Web-Front-End in vollem Funktionsumfang erwartet. Stattdessen sollen die essentiellen Funktionalitäten in einem Prototyp realisiert werden, um die verschiedenen und wichtigen Aspekte abzudecken und um die Machbarkeit zu demonstrieren und allfällige technische Probleme und Risikenaufzuzeigen. Insbesondere müssen nicht alle Parameter im UI abgedeckt werden, sondern ein paar wenige essentielle Parameter, um das UI Konzept und die Funktionsweise zu illustrieren. Weiter wird keine Visualisierung der Simulation erwartet. Diese wird gleich gehandhabt, wie beim bereits bestehenden User Interface der Ascomp. Nämlich werden die berechneten Resultatdateien mit einem externen Open Source Programm dargestellt. Dieses wird in einem späteren Kapitel beschrieben.

1.3 Vorgehen Aufbau der Arbeit

Die Arbeit ist in verschiedene Phasen aufgeteilt. Als erstes wird die bestehende Software der Firma Ascomp analysiert. Es wird basierend auf dem Workflow der Software eine Grobanalyse durchgeführt, in welcher die einzelnen erforderlichen und wünschenswerten Funktionen und Eigenschaften definiert werden. Danach wird ein Grobkonzept für die Architektur und Umsetzung dieser Funktionalität im Rahmen der Cloud- und Web-basierten Lösung erstellt. Dabei werden auch die technischen Risikopunkte identifiziert und separat aufgeführt und diskutiert. Diese Erkenntnisse werden in einer weiteren Phase priorisiert und detailliertere Lösungskonzepte erarbeitet.

Zur gleichen Zeit wird fortlaufend an einem Prototyp gearbeitet. Dieser wird, beginnend mit den Grundfunktionen für einen Architekturdurchstich, nach und nach um weitere Funktionen, die aus der Analyse ersichtlich werden, ergänzt. Besonders technische Risikopunkte werden implementiert, analysiert und soweit möglich gelöst. Ebenfalls wird zusammen mit dem Prototypen ein UI-Design erarbeitet, das kontinuierlich ergänzt und verbessert wird.

2 Stand der Technik

Im Folgenden werden die existierenden Hardware- und Software-Technologien und Lösungen im Bereich der Fluid Dynamik Simulation und einer entsprechenden Web-Architektur analysiert und erklärt. Zum Schluss werden allfällige Konkurrenzprodukte diskutiert, die dem Ziel dieser Bacherlorarbeit entsprechen oder damit verwandt sind.

2.1 Ascomp: TransAT

Die Software TransAT von der Firma Ascomp ist ein auf Linux entwickeltes Berechnungsprogramm. Es berechnet Simulationen im Bereich der Fluid Dynamik und gibt auch eine grafische Lösung der Simulation als Resultat zurück. Zu beachten ist dabei, dass für eine Berechnung in der Regel relativ grosse Input-Daten benötigt werden, diese können bis zu einem Gigabyte gross werden. Für die Berechnung einer Simulation können bis zu 1000 Eingabe-Parameter benötigt werden.

Das Programm setzt sich aus dem Berechnungsprogramm TransAT-Solver und aus der Benutzeroberfläche TransATUI, zusammen, welche in den nachfolgenden Kapiteln genauer beschrieben werden.

2.1.1 TransAT-Solver

Der TransAT-Solver ist der eigentliche Kern der TransAT-Software von Ascomp. Er ist zuständig für das Berechnen der Fluid Dynamik Simulationsprobleme.

Ein Berechnungs-Job besteht aus mehreren TransAT-Projektdateien. Der Solver nimmt diese Dateien entgegen und rechnet aufgrund der Angaben in diesen Dateien das Simulationsproblem. Als Resultat des Solvers werden verschiedene Lösungsdateien produziert. Auf die spezifischen Input- und Output-Dateien wird im nächsten Abschnitt eingegangen.

Aufgrund der sehr rechenintensiven Berechnungsaufgaben implementiert der Solver einen parallelen Berechnungsalgorithmus, so dass Multi-Cores und Multi-Prozessoren zur Beschleunigung der Laufzeit ausgenutzt werden. Dies kommt einer Portierung auf ein Cloud bzw. Cluster in hohem Masse entgegen, da sich die Simulation damit auf sehr vielen CPUs verteilen und parallel rechnen lässt.

2.1.1.1 TransAT-Solver-Projektdateien

Wie bereits im vorherigen Kapitel beschrieben, werden die Solver-Projektdateien benötigt, um dem Solver die relevanten Angaben für die Berechnung zu liefern. Diese Input-Dateien setzen sich aus dem Berechnungsgitter, den Objektoberflächen, den Domaingrenzen sowie den Simulationsparametern zusammen.

Auf die genauen Unterschiede der Dateien wird hier nicht weiter eingegangen. In der Tabelle 2.1 sind die einzelnen für den Solver relevanten Dateien aufgeführt. Zusätzlich werden ihre Funktionen grob beschrieben.

Dateiname	Funktion der Datei
Projektname.ls	Definiert das Berechnungsgitter über die Oberflächen eines Objekts. Die Datei kann sich aus mehreren CAD Dateien (z.B.*.gts, *.stl) zusammen-setzen.
properties.dat	Definiert die hauptsächlich physikalischen Eigenschaften jedes einzelnen Stoffes, z.B. Viskosität und spezifische Dichte.
Projektname.grda	Hier sind alle x-y-z-Koordinaten des Berechnungsgitters gespeichert.

Projektname.bc	Definiert die Systemgrenze der Simulation sowie den In- und Output in das System.
transat.inp	Speichert alle Simulationsparameter, z.B. wie viele Schritte gebraucht werden, wie die lange Simulation dauern soll, welches Schema verwendet werden soll, usw.
initialconditions.f90	Ein Fortran 90[1] Skript, welches die Anfangsbedingungen der TransAT-Simulation definiert. Das Skript kann in jedem beliebigen Texteditor bearbeitet werden. Das Skript wird kompiliert, zur Solver-Bibliothek gelinkt und ausgeführt. Des Weiteren hat das Skript Zugang zu allen Variablen und Arrays des Solvers. Dies macht dem Benutzer möglich, eine ideale Ausgangssituation für sein Projekt zu erstellen.

Tabelle 2.1: Solver Projektdateien

2.1.2 TransATUI

TransATUI ist die grafische Oberfläche, welche den Benutzer beim Projektmanagement sowie bei der Eingabe der bis zu 1000 Parameter unterstützt. Das UI ist so gestaltet, dass es den Benutzer durch den Arbeitsprozess, den sogenannten Workflow, führt. Auf diesen Workflow wird im nächsten Abschnitt näher eingegangen.

Grundsätzlich wäre für die Realisation eines TransAT-Projekts kein User Interface notwendig. Statt des TransATUIs kann ein Benutzer grundsätzlich auch mit der Konsole bzw. mit dem Texteditor arbeiten. Das heisst, die einzelnen Parameter und Bedingungen können auch explizit in die Projektdateien geschrieben werden, zumal diese Dateien auf einem ASCII Textformat und einer leicht editierbaren Textstruktur basieren. Zur Unterstützung des Workflows sowie zur Validierung von korrekten Eingaben oder zum Aufzeigen von Abhängigkeiten ist das User Interface jedoch nützlich. Ebenso erlaubt das TransATUI eine limitierte Visualisierung des Input-Gitters und des Simulationsobjektes.

Das TransATUI bietet zum jetzigen Zeitpunkt keine eingebettete Darstellung des Resultats vom Solver an. So ist es nicht möglich, die Resultate von Berechnungen direkt im UI zu simulieren und in 3D zu visualisieren. Für die Darstellung wird Paraview [2] verwendet, ein separates Open Source Programm, dass die Visualisation von simulierten Grafiken erlaubt.

2.1.2.1 TransATUI Workflow-Unterstützung

Die Abbildung 1 zeigt ein Screen Snapshot vom TransATUI. Darauf lässt sich erkennen, dass mit Hilfe von Tabs die Benutzeroberfläche in verschiedene Sektionen eingeteilt wird:

1. In der ersten Sektion wird das Projekt verwaltet. Es können Projekte neu erstellt, geladen und gespeichert oder ein bestehendes Projekt unter neuem Namen abgespeichert werden. Zusätzlich werden alle, dem Projekt zugehörigen, Projektdateien angezeigt. Die einzelnen Projekte werden vom Benutzer an einem beliebigen Ort auf dem Rechner abgelegt.
2. In der zweiten Sektion können alle Angaben und Parameter, die dem Setzen der Rahmenbedingung des Projektes dienen, eingegeben werden. Zusätzlich benötigte Dateien, die für das Einlesen einer Objektform dienen, jedoch letztendlich nicht vom Solver benötigt werden, können hier in das UI eingefügt werden.
3. Die dritte Sektion ermöglicht die Eingaben von allen Simulations-Parametern. Dazu gehören zum Beispiel die Definition der physikalischen Modelle, der Flüssigkeitseigenschaften und der

[1] http://de.wikipedia.org/wiki/Fortran
[2] http://www.paraview.org/

Simulationseigenschaften sowie die verschiedenen Konditionen der Simulation oder die Eigenschaften des Outputs, also des Resultats.

4. In der vierten und letzten Sektion wir der Befehl zur Berechnung des Projekts gegeben. Dabei werden die Daten vom TransATUI an den TransAT-Solver weitergeleitet. Eine fortlaufende Fortschrittsanzeige in Form eines Logs, wird in dem TransATUI angezeigt.

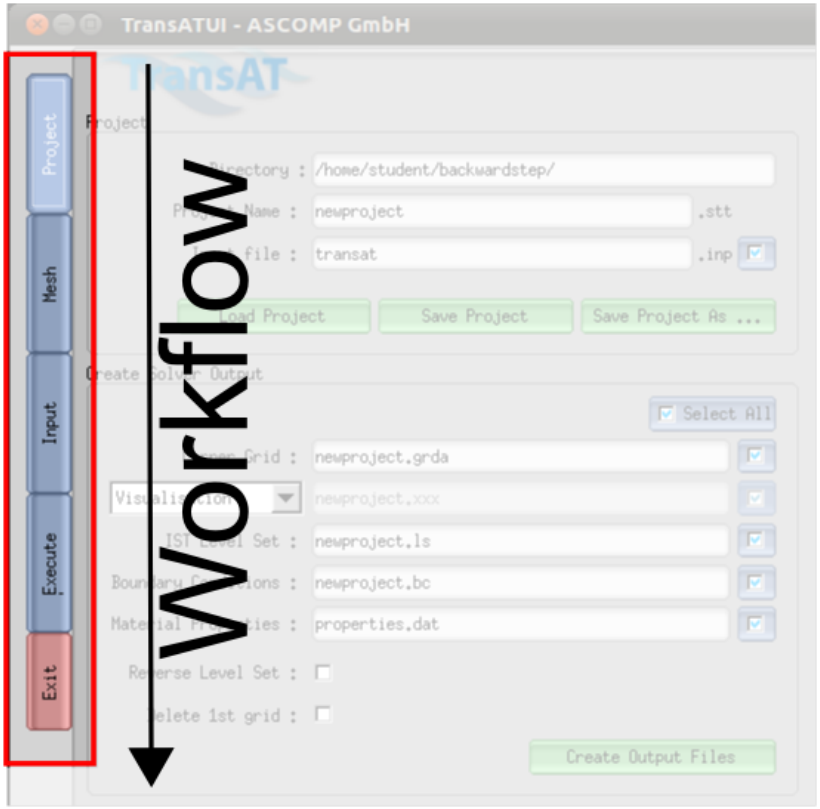

Abbildung 1: Workflowunterstützung im GUI

2.1.2.2 TransATUI-Projektdateien

In der Tabelle 2.1 sind die verschiedenen für das TransATUI relevanten Dateien aufgeführt und deren Zweck wird kurz beschrieben. Auf die genauen Unterschiede der Dateien wird nicht weiter eingegangen. Weitere Projektdateien, welche für den Solver relevant sind, werden aus den Parametereingaben im TransATUI erzeugt. Dateien, welche einzig für das TransATUI relevant sind, werden sinnvollerweise nicht an den Solver weitergeleitet.

*.gts	Ein internes Format, welches die Oberfläche und Geometrie der einzelnen Bauteile darstellt. Üblicherweise wird die Oberfläche mit einer Menge von Dreiecken, mit den jeweils zugehörigen X-Y-Z-Koordinaten, dargestellt. Dieses Format kann in fast alle gängige CAD-Software importiert werden.
*.stl	Beschreibt die Stoffoberflächengitter mit Dreieckskoordinaten. Dieses Format kann ebenfalls in fast alle gängige CAD-Software importiert werden.
transat.inp	Speichert alle Simulationsparameter, z.B. wie viele Schritte gebraucht werden, wie lange die Simulation dauern soll, welches Schema verwendet werden soll, usw.

initialconditions.f90	Ein Fortran 90[3] Skript, welches die Anfangsbedingungen der TransAT-Simulation definiert. Das Skript kann in jedem beliebigen Texteditor bearbeitet werden. Das Skript wird kompiliert, zur Solver-Bibliothek gelinkt und ausgeführt. Des Weiteren hat das Skript Zugang zu allen Variablen und Arrays des Solvers. Dies macht dem Benutzer möglich, eine ideale Ausgangssituation für sein Projekt zu erstellen.
projectname.stt	Archiv für alle TransATUI-Dateien sowie die UI-Parametereingaben zu den Rahmenbedingungen.

Tabelle 2.2: TransATUI -Projektdateien

2.1.2.3 Gts-Tool

Das Gts-Tool ist ein kleines Konsolen-basierendes Programm welches zur schnellen Erstellung von Geometrie-Objekten dient. Die daraus resultierenden Dateien (*.gts bzw. *.stl) können im TransAT-Solver verwendet werden.

2.1.3 Workflow mit TransATUI und TransAT Solver

In der Abbildung 2sieht man eine Visualisierung des Workflows mit den einzelnen Schritten und den jeweiligen daraus resultierenden Dateien.

Um die Grafik richtig interpretieren zu können, soll hier noch einmal darauf hingewiesen werden, dass die Applikation nicht auf das GUI angewiesen ist und deshalb die TransATUI- bzw. TransAT-Dateien in der Grafik aufgeteilt sind.

[3] http://de.wikipedia.org/wiki/Fortran

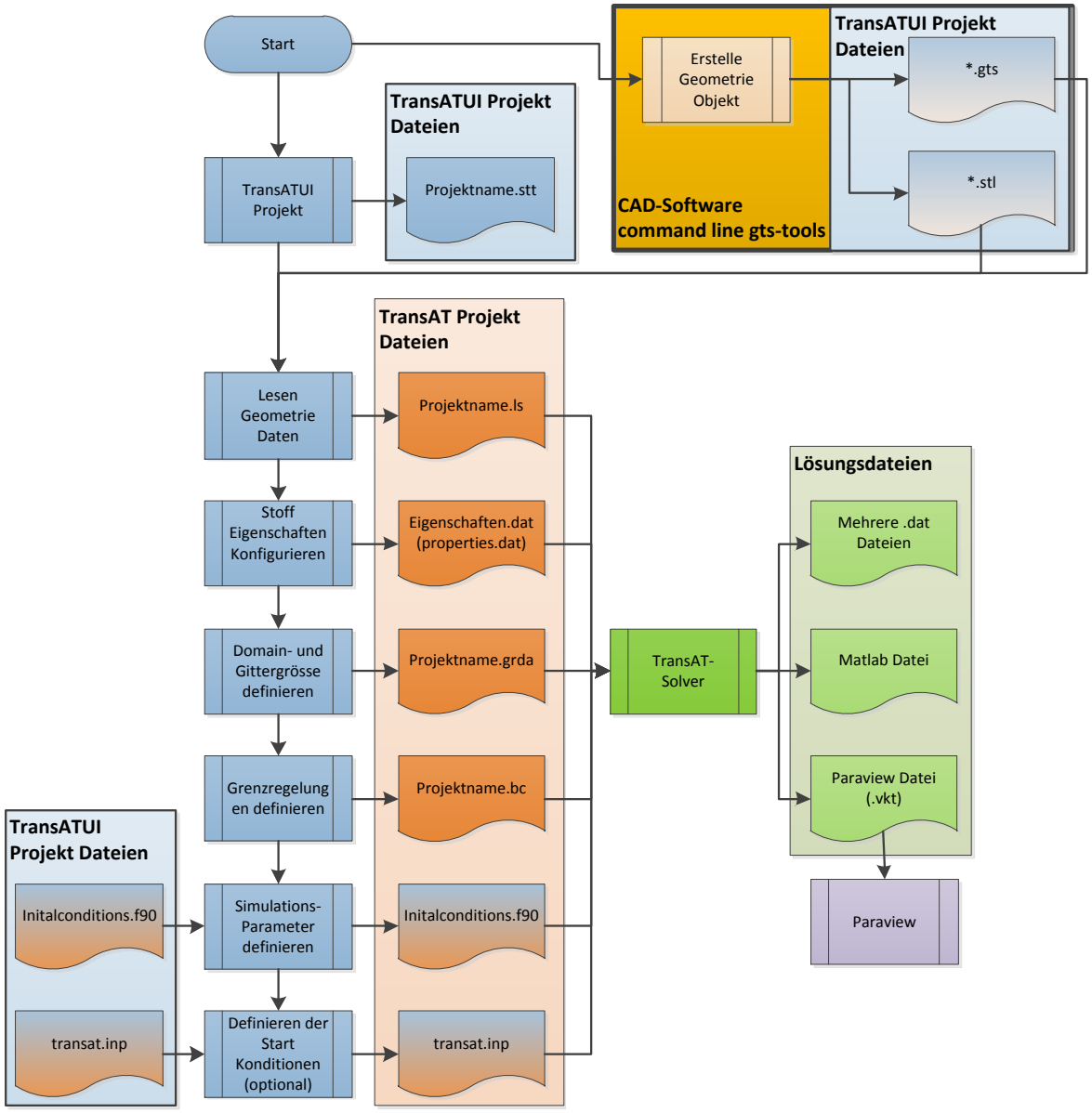

Abbildung 2: Workflow von TransATUI mit TransAT

2.2 HSR

Die HSR hat seit dem 7. Juni 2011 einen HPC-Cluster(High-Performance Computing Cluster) basierend auf Windows HPC Server 2008 R2.

Der Cluster besteht aus 33 sogenannten Rechenknoten mit 396 Kernen und 864 GB Ram. Die Rechner haben zusammengefasst eine Rechenleistung von 3163 Gigaflops. Die Cluster-Nodes sind untereinander mit einem Infiniband-Netzwerk verbunden. Dieses Netzwerk leistet theoretisch eine bidirektionale Übertragungsgeschwindigkeit von 2.5 GBit/s.

Der Cluster ist in das Active Directory der HSR eingebunden, was für eine Bachelorarbeit an der HSR durchaus seinen Reiz hat. Bis anhin wird der HPC noch nicht für externe Kunde der HSR verwendet. Dies führt dazu, dass die konkrete Schnittstelle von aussen zum Cluster noch nicht existiert bzw. noch nicht evaluiert worden ist. Die ist ein wichtiger Bereich, den es innerhalb dieser Arbeit abzuklären gilt.

2.3 Cloudbroker

Cloudbroker ist ein junges Dienstleistungsunternehmen, welches eine universelle Schnittstelle zu diversen Cloud-Infrastrukturen wie z.B. Amazon oder IBM bietet. Seit dem 2. November 2011 ist die Schnittstelle zur Infrastruktur von Cloudbroker einsatzbereit.

In der Abbildung 3 ist eine Übersicht über die Module von Cloudbroker abgebildet. Die Abbildung zeigt, dass alle wichtigen Aspekte einer Cloud-basierten Applikation abgedeckt werden. So sind z.B. Kosten- und Rechnungsmodule, Prozess Monitoring oder das User Management und noch vieles mehr vorhanden. Des Weiteren ist das TransAT System von ASCOMP auf Cloudbroker einsatzbereit.

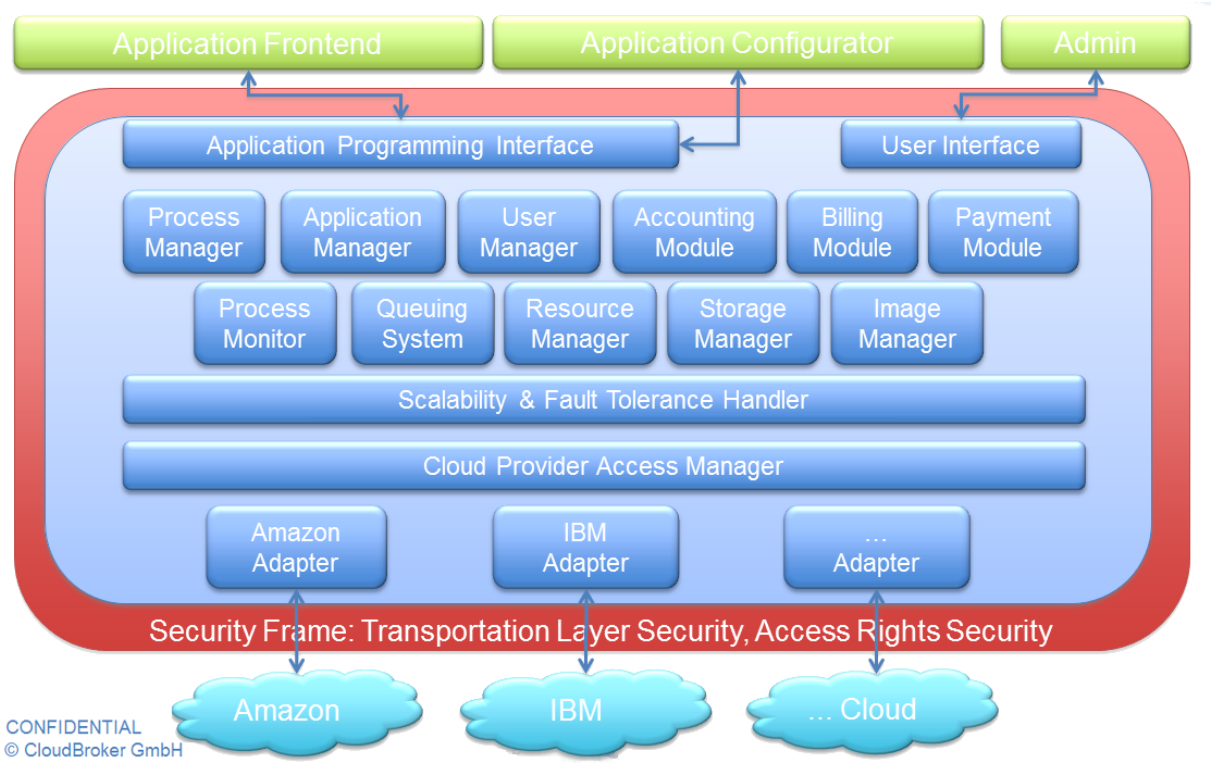

Abbildung 3: Cloudbroker Architektur

2.3.1 Zugang zur Infrastruktur

Cloudbroker ermöglicht zwei Varianten zur Kommunikation mit seiner Infrastruktur. Einerseits das Web-Interface, welches in der Abbildung 4 abgebildet ist sowie eine RESTful API, welche den Zugriff auf die Infrastruktur über einen REST-basierten Web Service erlaubt. Die Dokumentation der RESTful API ist noch nicht vollständig, jedoch wird sie laufend ergänzt. Von Cloudbroker noch nicht dokumentiert ist der für diese Arbeit relevante Teil über die Erstellung von Berechnungsaufträgen und das Auslesen des Status eines Auftrages.

Beide Varianten setzen einen Cloudbroker-Account voraus, an welchem die entsprechenden Softwares mit den entsprechenden Lizenzen, sowie die Abrechnung, gekoppelt ist. Somit stellt die Cloudbroker Achitektur, welche für diese Bachelorarbeit zur Verfügung steht die folgenden Module zur Verfügung:

- Process Manager, erstellen und starten der Aufträge
- Resourcen Manger, welche Resource stehen zur Verfügung
- Cloud Provider Acces Manger, ermöglicht Zugang zu den Clouds

Diese Module erlauben es, Berechnungsaufträge über die RESTful API zu erstellen und in einer der Clouds von Cloudbroker berechnen zu lassen.

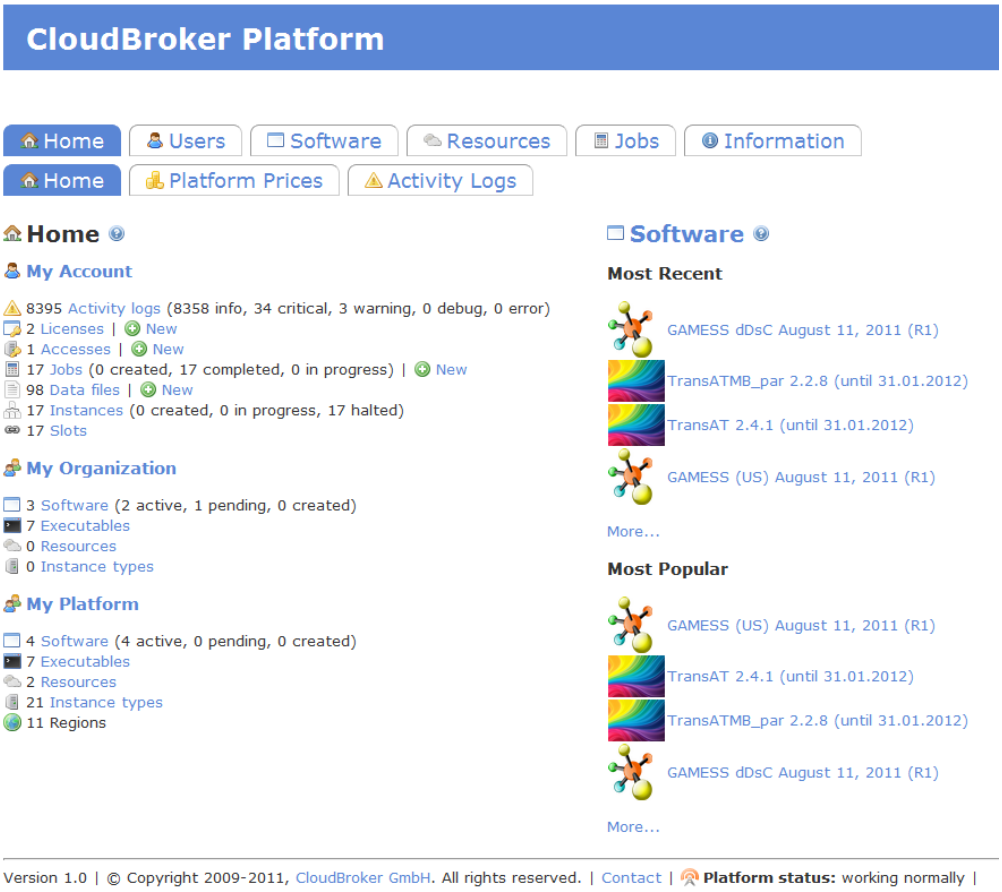

Abbildung 4: Cloudbroker Webinterface (19.12.2011)

2.4 Microsoft Silverlight 4

Microsoft Silverlight 4 ist ein Betriebssystem unabhängiges Browser-Plug-In, welches leistungsfähige Anwendungen mit ansprechender Benutzeroberfläche (sog. Rich Internet Application) im Internet ermöglicht. Eine Anwendung wird vom Webserver, auf dem sich der Quellcode der Anwendung befindet, heruntergeladen und auf dem Client ausgeführt. Der Client wird typischerweise durch einen Browser repräsentiert. Für die Verwendung einer Silverlight Applikation muss lediglich das Browser-Plug-In installiert werden.

2.5 Vergleichbare Konkurrenzprodukte

Es gibt zwar einige Konkurrenzprodukte für das TransAT, auch solche die ihre Berechnungen in der Cloud machen, aber unsere Recherchen haben kein vollständig web-basiertes Simulationssystem ergeben, dass keine Installation beim Clienten erfordert (sog. „No-Installation System“).

3 Evaluation

Dieses Kapitel dokumentiert alle wesentlichen Entscheidungen und Konzepte, welche während der Arbeit getroffen bzw. erarbeitet wurden. Weiter werden Risiken und Schwierigkeiten aufgelistet, welche im Verlauf des Projekts aufgetreten sind.

In einem ersten Teil wird der Bereich der Analyse des Projekts evaluiert, während in einem zweiten Teil Entscheidungen und Konzepte sowie Risiken und Schwierigkeiten, welche technischer Natur sind, evaluiert werden. Entscheidungen, welche für die Entwicklung des Prototypens getroffen wurden, werden im Kapitel Prototypen der Software Dokumentation beschrieben.

3.1 Analyse

Bei der Analyse des Projekts sind die folgenden Punkte als sehr wichtig eingestuft worden:

- Anbindung des WebUI Front-Ends an ein Cloud-Backend, welche die Simulation mit dem TransAT Solver erlaubt
- Unterstützung des Workflows (Arbeitsflusses) im UI

3.1.1 Cloud-Backend

Die Anbindung an ein Cloud-Backend, welches die Simulationsprojekte aus dem WebUI Front-End entgegennimmt, diese berechnet und wieder an das Front-End zurück gibt, ist ein wesentlicher Bestandteil der Machbarkeitsstudie. Ohne diese Funktion kann die Machbarkeit nicht geprüft werden, da ein wichtiger Teil der Simulation fehlen würde und diese nicht durchgehend automatisch ablaufen könnte.

3.1.2 UI Workflow

Bei dem Erstellen eines Simulationsprojekts ist ein schrittweiser Arbeitsablauf vorgesehen, welcher die Arbeit erleichtert und ein versehentliches Absenden des Simulationsprojekts an den Solver verhindert, da dessen Berechnung sehr lange dauern kann und, falls er sich in einer Cloud befindet, die Berechnung nicht kostenlos ist. Der Vorgang besteht grob aus den folgenden Punkten:

1. Erstellen des Projekt
2. Bearbeiten der Projektdateien (diese sind evtl. gleich wie in einem anderen Projekt und können nachgezogen werden)
3. Eingabe der Parameter, welche das Umfeld der Simulation definieren
4. Eingabe der Parameter, welche die Simulation selbst definieren
5. Bestätigung der Eingaben
6. Berechnungsbefehl an den Solver
7. Entgegennehmen der Lösung

Auf diesen Workflow soll auch bei der Umsetzung des WebUI Front-Ends ein grosser Wert gelegt werden. Der Benutzer so gut wie möglich mit diesem Workflow unterstützt werden.

3.2 Entscheidungen

3.2.1 Mockup des HPC

Am 9. November 2011 wurde erkannt, dass die von anderen Entwicklern geplante Portierung von TransAT auf den Windows HPC nicht mehr während der Dauer der Bachelorarbeit stattfinden wird.

Weiter war auch Cloudbroker noch nicht vollständig einsatzbereit. Weiter war auch die Machbarkeit mit Cloudbroker noch nicht sicher. Aus diesem Grund wurde entschieden, vorübergehend eine Mockup-Lösung für den HPC zu schreiben, der diesen zusammen mit der TransAT-Berechnung simuliert. Auf diese Weise konnte der Prototyp vorübergehend ohne Einschränkungen weiterentwickelt werden.

3.2.2 Priorisierung

Die Priorisierung fand am 23. November 2011 statt, als bei der Entwicklung des Front-Ends der Zwischenstand erreicht wurde, bei dem alle wichtigen Funktionen, wie z.B. Projekt erstellen, reduzierte Parameterbearbeitung, Senden des Berechnungsauftrags und Empfangen der Lösungsdatei, implementiert waren. Weiter war die Schnittstelle zum HPC zwar fertig, jedoch ohne Aussicht dass TransAT darauf zu laufen kommt. Dafür war die Schnittstelle von Cloudbroker seit kurzem einsatzbereit und weiter war auch TransAT einsatzbereit auf Cloudbroker.

Für den weiteren Verlauf der Bachelorarbeit standen noch etwa 130 Stunden für die Entwicklung des Prototyps zur Verfügung. Somit wurden die in der Tabelle 3.1 aufgelisteten, nach Priorität geordneten, Vertiefungspunkte für die Weiterentwicklung des Prototyps in Betracht gezogen.

Vertiefungspunkt	Beschreibung	Geschätzter Zeitaufwand
Backend (1. Priorität)	Anbindung an das Cloudbroker-Interface um den Solver in das Projekt einzubinden.	60 h
UI Workflow (2. Priorität)	Erstellung und Umsetzung eines Konzepts für das UI, das die Usability verbessert. So soll das UI z.B. einen besseren schrittweisen Ablauf vorweisen oder Statusanzeigen enthalten usw.	50 h
Parameter (3. Priorität)	Der Parser, der die Parameter in die Projektdateien liest oder schreibt soll ausgebaut werden. Weiter soll dafür ein Erweiterungskonzept erstellt werden, damit die Parameter flexibel erweiterbar sind.	30 h
Job Monitoring (4. Priorität)	Feedback von der Cloudschnittstelle um den Fortschritt einer laufenden Berechnung zu erhalten.	30 h
Benutzerverwaltung (5. Priorität)	- Benutzerdatenbank - Zugriff nur für rechtmässige Benutzer (Login) - Unterscheidung/Zuweisung der Projektfiles im Cluster - Rückgabe der Resultatfiles an richtigen Client	30 h
Projektverwaltung (6. Priorität)	- Projektablage - Persistieren von Projektmetadaten - Ablage der Projektdateien auf einem Fileserver	50 h

Tabelle 3.1: Vertiefungspunkte

Die Vertiefungspunkte wurden zusammen mit dem Betreuer und der Partnerfirma diskutiert. Es wurde entschieden, dass die Vertiefungspunkte 1 + 2, also die Anbindung an Cloudbroker sowie die Erweiterung des UIs umgesetzt werden sollen. Alle weiteren Vertiefungspunkte sollen vernachlässigt werden.

3.3 Konzepte

3.3.1 Schnittstelle zum Cloud-Backend

Zu Beginn des Projekts wurde das Konzept einer einheitlichen Schnittstelle zum Cloud-Backend festgelegt. Das heisst, das WebUI Front-End wird unabhängig von der Cloud entwickelt. Dies

ermöglicht den Einsatz einer beliebigen Cloud-Lösung im Backend, ohne dass dabei im Front-End Anpassungen benötigt werden. Einzig auf der Seite des Silverlight Webservers müssen die Web Services, welche die Funktionen der Schnittstelle zum Cloud-Backend implementieren, angepasst werden.

3.4 Risiken und Schwierigkeiten

Die Hauptrisiken und Schwierigkeiten lagen vor allem an Netzwerk und an Verspätungen der Projektpartner. Mit den Risiken wurden zwar von Anfang an gerechnet, dass sie aber in diesem Ausmass eingetreten sind, war nicht vorausgesehen. Die erste Schwierigkeit lag beim HPC, dessen Infrastruktur nicht unseren Anforderungen entsprach. Die Verspätung von Cloudbroker liess es jedoch am Anfang nicht zu, auf ein anderes Cloud-Backend umzustellen. Dies führte dazu, dass eine eigene Schnittstelle für den HPC implementiert wurde. Die Implementation der Schnittstelle erwies sich als nicht ganz einfach, war jedoch zu bewältigen. Doch bereits traf auch schon wieder das nächste Risiko ein. Die von anderen Entwicklern geplante Portierung des TransAT Systems auf Windows wurde auf einen unbekannten Zeitpunkt aufgeschoben. Nach dem Erstellen eines Mock-ups wurde dann jedoch das Cloudbroker-Backend einsatzbereit und ermöglichte letztendlich doch noch die vollständige und lauffähige Implementation eines Cloud-Backends.

Bei der Entwicklung des Front-Ends tauchten fast keine Schwierigkeiten auf. Bei den Risiken ist einzig eingetreten, dass die Silverlight-Applikation nicht auf allen Plattformen lauffähig ist. Leider war das Beheben dieses Mangels nicht möglich, da dessen Auftreten dem Silverlight-Plugin zu belasten ist. Als Schwierigkeiten sind beim Front-End die Silverlight-Einschränkungen aufgetreten. Die Silverlight-Applikation läuft in einer Sandbox. Das heisst, es kann nur sehr beschränkt auf die Ressourcen des Benutzerrechners zugegriffen werden. Es gibt Möglichkeiten, wie z.B. der Isolated Storage für die Fileablage, mit welchen gewisse Einschränkungen in sehr begrenzter Weise umgehen werden können. Leider sind diese Möglichkeiten häufig sehr kompliziert und erfordern gute Kenntnisse. Die Implementation und die vorgängige Einarbeitung in die Funktionen des Isolatet Storages nahm aus diesem Grund mehr Zeit in Anspruch als ursprünglich geplant war.

4 Resultate

Die zum Ende des Projekts erreichten Ziele sowie persönliche Erkenntnisse und daraus folgende Schlussfolgerungen werden in diesem Kapitel beschrieben.

4.1 Erreichte Ziele

Das Ziel dieser Bachelorarbeit: Die Machbarkeitstudie zum Thema „WebUI Front-End for Fluid Dynamics Cloud Computing“ wurde erfüllt. Es ist gelungen, die Machbarkeit in Form eines Prototyps zu beweisen. Der Prototyp deckt dabei die grundlegenden Funktionen des gewünschten Endobjekts ab. Die Details des Resultats sind in der Software Dokumentation unter dem Kapitel Resultate aufgelistet.

4.2 Persönliche Berichte

4.2.1 Patrice Müller

4.2.1.1 Projektverlauf

Für mich stand schon von Anfang an fest, dass ich etwas mit C# und dem .Net-Framework machen will. Denn ich wollte noch eine weitere Technologie kennenlernen. Da ich, wie die meisten, bis anhin hauptsächlich mit Java gearbeitet habe. Zwar hatte ich einen kleinen Einblick in die Welt von C# und .Net im Modul „MS-Tech“, aber ein reales Projekt konnte ich bis jetzt nicht vorweisen. Des Weiteren sollte die Arbeit nicht unbedingt etwas mit Mobile zu tun haben, da ich mich mit diesem Thema schon in meiner Semesterarbeit auseinandergesetzt habe.

Als ich dann die ausgeschriebene Silverlight-Arbeit „Web UI Front-End für Fluid Dynamics Cloud Computing“ sah, war ich gleich begeistert. Denn ich sah die Chance, mich mit C# und der momentan hoch diskutierten Thematik Cloud-Computing auseinanderzusetzen. Und mit Frau Hollenstein fand ich eine kompetente und angenehme Projektpartnerin.

Eine Woche vor Semesterbeginn war dann das Kick-Off-Meeting bei der Ascomp und uns wurde TransAt vorgestellt. Ein hoch komplexes Programm mit bis zu 1000 Parametern. Des Weiteren erfuhren wir dort, dass die Portierung des Programms TransAT auf Windows noch ausstehend ist, diese aber in den nächsten paar Wochen nachgeliefert wird, damit wir auf dem HPC arbeiten können. Dazu erfuhren wir auch noch, dass es neben der HSR, noch einen zweiten Cloud-Anbieter gab und zwar die Firma Cloudbroker.

Auf der Infrastruktur von Cloudbroker würde zwar das Programm TransAT schon laufen aber uns wurde dann gleich gesagt, das Cloudbroker noch nicht ganz fertig sei mit ihrer Entwicklung der Schnittstelle. Somit stand fest, dass unser Fokus auf dem HPC der HSR liegt. Aber bevor wir uns mit der Thematik Cloud auseinandersetzten, wollten wir uns erst einmal um die Analyse des Ascomp Programms TransAT kümmern.

Die Analyse war ein hochspannender Prozess, denn wir durften einen kleinen Blick in die Welt der Fluid Dynamik Simulationssystem werfen und deren verwendeten Sprachen wie fortran90[4] kennen lernen. Am Ende dieses Prozesses waren wir zwar immer noch weit davon entfernt, alles zu verstehen, was das Programm macht, geschweige denn, wie man ein Simulation von Grund an selber konfiguriert. Dennoch wussten wir, welche Dateien für was sind und welche zwingend nötig sind um

4

ein Projekt, sprich eine Simulation, durchzurechnen. Dies waren genau die Informationen, die wir für die Erfüllung unserer Aufgabe brauchten.

Jetzt konnten wir uns um die Cloud kümmern. Da wie schon gesagt die Firma Cloudbroker noch mit der Fertigstellung ihrer Schnittelle beschäftig war, konzentrierten wir uns auf den HPC der HSR. Nach anfänglicher Freude, dass der Zugang so schnell klappte (Active Directory der HSR), mussten wir schnell feststellen, dass der HPC alles andere als bereit war externe Kunden zu „bedienen“. Denn der HPC der HSR war von aussen nicht erreichbar, also eigentlich unbrauchbar für unseren Kunden Ascomp.

Dennoch gaben wir nicht auf. Wir richteten einen Virtuellen-Proxy in der DMZ ein, welcher von da an einerseits unser Webserver war, um unsere Silverlight-Applikation zu hosten und andererseits uns den Datenaustausch mit dem HPC ermöglichte.

Über eine Windowsfreigabe auf dem Proxy, welche vom HPC überwacht wurde, wurde der Datenaustausch bewerkstelligt. Der dazugehörige Windowsservice wurde von uns geschrieben.

Leider half dies alles nichts, da die noch ausstehende Portierung des Programms TransAt, welche zwingend nötig für unser Weiterkommen war, ausblieb und die Schnittstelle von Cloudbroker auch noch nicht fertig war. So schrieben wir uns einen HPC-Mock-up welcher die Weiterentwicklung unserer Silverlight-Applikation garantierte. Leider sind der Mock-up und der Windowsservice Wegwerfprodukte unser Arbeit.

Als uns dann in der zweiten Hälfte der Bachelorarbeit die Nachricht erreichte, dass Cloudbroker mit der Schnittstelle nun fertig sei, schwenkten wir zu Cloudbroker um.

Nach anfänglichen Schwierigkeiten, welche mehr oder weniger zu erwarten waren, konnten wir bei einer 1.0 Version der Cloudbroker-Schnittstelle einen lauffähigen Prototyp fertigstellen.

4.2.1.2 Persönlicher Rückblick

Die Arbeit hatte es in sich. Neben der komplexen Materie der Fluid Dynamik und den neuen Technologien (C#, Silverlight, WPF und allg. Clouds), gab es noch einige Rückschläge die wir erleben durften.

Aber genau durch diese Rückschläge, musste ich mich auch mit Themen auseinandersetzen mit denen ich sonst nie in Berührung gekommen wäre, wie z.B. Windowsservices.

Das spannendsten an dieser Arbeit fand ich, jedoch nicht die Technologien, sondern das Zusammenwirken so vieler Parteien. Die Hilfsbereitschaft vom HPC-Team und vom Cloudbroker-Team empfand ich als sehr angenehm. Gerade die Diskussionen mit diesen Teams fand ich sehr bereichernd.

4.2.1.3 Fazit

Es war eine sehr spannende Arbeit, aus der man mehr als nur Technisches-Know-How mitnehmen kann. Es wurde mir vor die Augen geführt, was es heisst, mit mehreren Teams aus mehreren Firmen zusammen zu arbeiten. Und wie man sich den Staub von den Schultern klopft nach mehr als einem Rückschlag und sagen kann: „Jetzt erst recht!“. Kurz gesagt; Ich freue mich jetzt schon auf die nächste Herausforderung.

4.2.2 Anita Hollenstein

4.2.2.1 Allgemein

Mit meiner Studienarbeit konnte ich bereits erste Erfahrungen mit .Net und Silverlight machen. Die Silverlight-Applikation der Studienarbeit hat sich jedoch mehr auf das Darstellen von Daten konzentriert, während bei der Bachelorarbeit ein User Interface im Vordergrund lag. Die Möglichkeit meine Kenntnisse in Silverlight und .Net zu vertiefen, wie auch das Ganze mit einem Cloud-Backend in Verbindung zu bringen, hat mich sehr gereizt an dieser Arbeit.

4.2.2.2 Projektverlauf

Zu Beginn des Projekts ging es als erst einmal darum, das Fluid Dynamik Berechnungsprogramm TransAT der Firma ASCOMP kennenzulernen. Als erstes war ich von den unzähligen Parametern „erschlagen“. Das Programm war inhaltlich sehr schwer zu verstehen, da natürlich Kenntnisse der Fluid Dynamik für die Anwendung des Programms vorausgesetzt sind, welche ich nicht besitze. Während der Analyse wurden jedoch das Programm, oder zumindest die einzelnen Abläufe davon, immer verständlicher. Mit der Analyse konnten die wichtigsten Funktionen und Fakten des Programms erarbeitet werden. Somit konnten folgend alle wichtigen Funktionen und wünschenswerten Eigenschaften der Machbarkeitsstudie festgelegt werden.

Während ich mich nach der Analyse um die ersten Funktionalitäten im Silverlight Client kümmerte, begab sich Patrice Müller daran, die Netzwerkinfrastruktur für den HPC aufzubauen. Während der Dauer des Projekts hat es sich dann automatisch ergeben, dass Patrice Müller für die Cloud-Schnittstelle und das Cloud-Backend zuständig war, während ich mich um Teil der Silverlight-Applikation kümmerte. So beschäftigte ich mich während der Bachelorarbeit grösstenteils mit der fortlaufenden Entwicklung des UIs und der zugehörigen Logik.

4.2.2.3 Fazit

Die Bachelorarbeit war für mich eine sehr gute Erfahrung. Ich habe dabei sehr viel gelernt. Es war eine sehr spannende Arbeit, die auch zeigte, wie es normalerweise bei Softwareprojekten läuft. Es gab nämlich häufige Verspätungen und Wartezeiten die unter anderem auch aufgrund der vielen Beteiligten im Projekt aufgetreten sind.

Natürlich waren diese Verspätungen und Wartezeiten auch eine ziemliche Belastung. Denn in der ohnehin schon kurzen Projektdauer von 14 Wochen hat jede noch so kurze Verzögerung grosse Auswirkungen auf das Endresultat. So ist es zum Beispiel schade, dass nicht bereits von Anfang an das Cloudbrocker Cloud-Backend verwendet werden konnte. Dieses stellt nämlich eine Vielzahl an Funktionen dar, schon von Anfang an in das UI eingeplant werden hätten können. Des Weiteren wäre es auch möglich gewesen, dass nicht nur ich grösstenteils an dem UI gearbeitet hätte, sondern auch Patrice Müller einen Teil des UIs übernommen hätte. Denn mit gemeinsamer Arbeit, wäre das UI bestimmt noch umfangreicher geworden. Leider nahm jedoch auch das Backend aufgrund der vielen Änderungen fast die ganzen 14 Wochen Zeit in Anspruch.

Es gäbe sicherlich noch sehr viele Ideen das UI zu erweitern. Teilweise fehlt auch nur noch die Arbeit von ca. 1 bis 2 Wochen um gewisse Funktionen, wie zum Beispiel der Abbruch einer laufenden Funktion oder das Anzeigen eines Berechnungslogs, zu erstellen. Aus diesem Grund finde ich es fast schade, das Projekt bereits nach 14 Wochen abzugeben. Andererseits erforderte das Projekt sehr viel Arbeitsstunden und ich bin froh, wenn es dann nach der Abgabe wieder etwas ruhiger zu und hergeht.

Im Projekt haben Patrice Müller und ich einander sehr gut ergänzt. Wir konnten die Arbeit unseren Interessen gerecht und gleichmässig aufteilen. Weiter war es uns aber auch möglich gemeinsam gut zu arbeiten.

4.3 Lessons Learned

Das Team hat aus dem Projekt folgende Punkte gelernt:

- **Cloud-Computing**: Das Betreiben einer Cloud ist recht aufwändig und muss sehr gut durchdacht sein, insbesondere, wenn man sie externen Kunden zur Verfügung stellen will.
- Die **Zusammenarbeit mit Partner von Kunden** braucht manchmal viel Zeit bis man mit der richtigen Person redet, ist dann aber meistens sehr produktiv.
- **Testing:** Das Testen von Silverlight-Applikation ist schwierig, dafür erwies sich das Testen von Webservices mit Visual Studio 2010 als ziemlich einfach.
- **XMLSerialisierung :** Das Serialisieren von Objekte zu XML und wieder zurück war überraschend einfach.

Teil II

Software Projektdokumentation

1 Anforderungsspezifikation

Beim WebUI-Front-End soll es sich um eine web-basierte Lösung handeln. Dieses Kapitel beschreibt die Motivation, diese web-basierte Lösung einzusetzen sowie die Anforderungen welche die verschiedenen Aktoren an das WebUI-Front-End stellen.

1.1 Motivation

Für den Einsatz einer web-basierten Lösung sprechen diverse Vorteile. Diese ermöglicht zum Beispiel eine Verwendung der Applikation ohne diese vorher installieren zu müssen. Die Benutzung der Applikation erfolgt über einen Webbrowser wie zum Beispiel Firefox oder Internet Explorer. Unter Umständen muss je nach Lösung ein zusätzliches Plugin für den Browser installiert werden, dies erfolgt jedoch direkt im Browser und erfordert lediglich eine Bestätigung des Benutzers, dieses Plugin installieren zu dürfen. Zu beachten ist, dass das WebUI-Front-End eine öffentliche Webseite darstellt, auf welche jeder zugreifen kann. Dies setzt voraus, dass die Benutzung der Applikation eingeschränkt wird, in dem zum Beispiel nur mit einem Login der volle Zugriff auf die Applikation gewährt wird.

1.2 Grundlage

Als Grundlage der Anforderungsspezifikation dient die bestehende Benutzeroberfläche, das TransATUI. Dabei wird der Fokus vor allem auf den Workflow des TransATUI gelegt. Da die bisherige Lösung eine komplett lokale Anwendung war, bestehen weitere Anforderungen an das Web Front-End. Aspekte wie die Erreichbarkeit oder die Multi-User-Datenverwaltung erhalten bei der Webbasierten Lösung eine zentrale Rolle.

1.3 Anforderungen

Im Folgenden werden die bisherigen Anforderungen beschrieben, welche bereits aus dem TransATUI bekannt sind. Danach wird der Fokus auf neue Anforderungen gelegt, die bei der bisherigen Anwendung noch nicht benötigt wurden.

1.3.1 Bisherige Anforderungen

Zu den bisherigen Anforderungen gehören die folgenden Punkte:

- Erstellen von Simulationsprojekten
- Bearbeitung eines Simulationsprojektes mit der Eingabe von bis zu 1000 Parameter
- Mapping der Parameter eines geöffneten Simulationsprojekts auf das UI
- Mapping der UI-Parametereingaben auf das Simulationsprojekt, bzw. dessen Projektdateien. Dies erfolgt über einen expliziten Befehl des Benutzers.
- Berechnung des Simulationsprojektes anhand der eingegebenen Parameter
- Abbruch der Berechnung des Simulationsprojektes
- Anzeige des Berechnungslogs um den Fortschritt der Berechnung zu verfolgen
- Anzeige des Resultats mit einem externen Visualisierungsprogramm

1.3.2 Neue Anforderungen

In der bisherigen Lösung wurden die Projekte lokal abgespeichert. Da das WebUI jedoch lokal unabhängig sein soll, wird neu eine Projektverwaltung erforderlich. Die Projekte müssen auf einer Datenbank oder auf dem Webserver abgespeichert werden können, damit sie lokal unabhängig verwendet werden können.

Weiter stand das Programm bisher unter Kopierschutz. Neu ist es für alle im Internet zugänglich. Trotzdem sollten nur Benutzer zugelassen werden, die über eine Lizenz für die Verwendung der TransAT Software verfügen. Das Programm soll also neu über eine Benutzerverwaltung verfügen und somit durch Authentisierung der Benutzer, also durch einen Login, nur zugelassenen Benutzer den Zugriff auf das WebUI gewähren.

1.4 Nicht funktionale Anforderungen

1.4.1 Zuverlässigkeit

Der Workflow soll bei einem allfälligen Verbindungsunterbruch nicht auch unterbrochen werden, sondern soll nach dem Wiederherstellen der Verbindung an der gleichen Stelle fortgesetzt werden können.

1.4.2 Benutzbarkeit

Trotz der sehr vielen Parameter, die in der Benutzeroberfläche angegeben werden können oder müssen, soll diese verständlich und einfach zu bedienen sein. Dies wird einerseits durch den Workflow unterstützt. Weiter sollen die Parameter so gruppiert werden, dass sie leichter zu finden oder ihre Funktion aufgrund der Gruppierung nachvollzogen werden kann. Weiter sollen durch Validation Falscheingaben überprüft werden. Wenn ein Parameter von einem anderen Parameter abhängig ist, so soll dieser solange inaktiv sein, bis der erste Parameter einen Wert erhalten hat.

1.4.3 Leistung und Effizienz

1.4.3.1 Antwortzeiten

Trotz der grossen Datenmengen, die bis zu einem Gigabyte gross sein können, dürfen die Antwortzeiten nicht übermässig lange sein. Leider lässt es sich kaum vermeiden, dass die Übertragung der Daten eher lange dauert. Eine genaue Fortschrittsangabe lässt sich nur schlecht realisieren, da jeder Job aufgrund seiner Grösse aber auch aufgrund der Simulationsrechenzeiten sowie der Auslastung des Netzes oder der Cloud bzw. dem Cluster verschieden lange dauern kann. Es soll jedoch immer eine Fortschrittsanzeige oder ein Status dargestellt werden, damit der Benutzer eine Rückmeldung zur laufenden Operation hat. Weiter soll es auch möglich sein, Operationen, die länger als 5 Sekunden dauern, abbrechen zu können.

1.4.3.2 Ressourcenbedarf

Der Silverlight-Client beansprucht die Ressourcen des Client-Rechners. Aus diesem Grund sollen Aufgaben mit hohem Rechner-Ressourcenbedarf möglichst auf den Cluster ausgelagert werden. Zusätzlich soll auch möglichst nur so viel Speicher des Clients wie nötig benutzt werden. Das bedeutet, dass Projekte, die im Moment nicht bearbeitet werden, z.B. auf eine Datenbank ausgelagert werden sollen. Konkret heisst das, dass mindestens ein Projekt auf dem Client gespeichert werden kann. Dieses nimmt einen Platz von maximal einem Gigabyte ein.

1.4.4 Wartbarkeit, Änderbarkeit

Es ist bekannt, dass die Benutzeroberfläche häufige Änderungen wahrnehmen soll. Das heisst es müssen neue Parameter hinzugefügt, geändert und entfernt werden können. Aus diesem Grund soll das Design der Benutzeroberflächen eine flexible und dynamische Änderbar- und Erweiterbarkeit vorweisen.

1.4.5 Portierbarkeit und Übertragbarkeit

Das Programm soll auf verschiedenen Plattformen lauffähig sein. Zu den unterstützten Plattformen gehören Windows, Mac OS und Linux. Dabei muss beachtet werden, dass für Linux Moonlight verwendet wird. Hierbei handelt es sich um eine quelloffene Implementierung von Silverlight.

1.4.6 Sicherheitsanforderungen

1.4.6.1 Vertraulichkeit und Datenintegrität

Im Solver wird eine beliebige Abfolge von Berechnungsaufträgen von Simulationsprojekten abgearbeitet. Diese Aufträge stammen von verschiedenen Benutzern. Weiter können auch mehrere verschiedene Aufträge von einem einzelnen Benutzer stammen. Die Aufträge müssen also eindeutig voneinander unterschieden werden können, damit die Resultate letztendlich an den richtigen Benutzer gelangen. Dies kann erreicht werden, indem die Benutzer durch Authentisierung eindeutig voneinander unterschieden werden. Weiter müssen die Benutzer dem Solver bzw. dem Cloud-/Cluster-Anbieter bekannt sein, damit er die Aufträge separieren kann. Dies könnte z.B. durch getrennte Working Directorys erfolgen.

1.5 Use Cases

1.5.1 Akteure & Stakeholders

1.5.1.1 Primäre Akteure

- Benutzer:
  Kunde der Ascomp, welcher das WebUI benutzt.

1.5.1.2 Stakeholders

- Entwickler:
  Mitarbeiter der Ascomp, welcher die Programmfunktionen auf dem Laufenden hält.
- Cloud-Anbieter:
  Stellt dem WebUI ein Interface für die Berechnung mit dem in der Cloud befindlichen Solver zur Verfügung.

1.5.2 Szenarios

Die Firma Y möchte mit dem TransAT Solver der Firma Ascomp ein Problem der Fluid Dynamik berechnen. Leider verfügt Y nicht über die benötigten Rechenressourcen des TransAT Solvers und musste aus diesem Grund bei der bisherigen lokalen Ausführung des TransAT(UI) Programms immer sehr lange auf ein Resultat warten.

Über das TransAT WebUI kann die Firma Y TransAT Projekte verwalten und bearbeiten und diese zur Berechnung an den TransAT Solver senden, der sich in einer Cloud befindet. Ist das Projekt fertig berechnet, erhält die Firma Y die berechnete Lösung und kann diese über das WebUI herunterladen.

Somit erspart sich die Firma Y sowohl Rechenressourcen für die Berechnung des Problems als auch Platzressourcen für die Aufbewahrung und die Verwaltung von Projekten.

Die Projekte, welche sich noch vom vorherigen Gebrauch des lokalen Programms auf den Laufwerken der Firma Y befinden, können problemlos in die Projektverwaltung des WebUI's hochgeladen werden.

1.5.3 Use Cases im Brief Format

Im Folgenden wird ein kurzer Überblick über alle Use Cases gegeben, welche für den Primär Akteur, also den Benutzer von Bedeutung sind.

1.5.3.1 UC01: Simulationsprojekte anlegen und verwalten

Der Benutzer kann seine Simulationsprojekte verwalten. Das heisst, für ein Projekt können die CRUD-Funktionen angewendet werden. Die CRUD-Funktionen (Create, Read, Update, Delete) setzten sich aus „Erstellen“, „Lesen“, „Bearbeiten“ und „Löschen“ zusammen.

Weiter hat der Benutzer die Möglichkeit, auch lokal vorhandene Projekte oder Projektdateien über das WebUI an den Solver zu senden bzw. dieses in die Projektverwaltung aufzunehmen. Aus diesem Grunde besteht die Möglichkeit ein lokales Projekt oder einzelne Dateien davon auf das WebUI hochzuladen. Ebenso können im WebUI vorhandene Projekte und Projektdateien auch auf den lokalen Rechner heruntergeladen werden.

1.5.3.2 UC02: Parameter bearbeiten

Die Parametereingaben, welche der Benutzer in einem Projekt macht, werden in verschiedenen Projektdateien abgelegt. Indirekt wird somit nicht das Projekt selber, sondern die dem Projekt zugehörigen Dateien bearbeitet. Die Parameter werden mittels Parser in einem bestimmten Format in eine ASCII-Datei hineingeschrieben oder ausgelesen.

1.5.3.3 UC03: Job an Solver senden

Ist die Bearbeitung eines Projekts fertig, so kann der Benutzer dieses zur Berechnung an den Solver absenden. Ein zu berechnendes Projekt wird Job genannt.

1.5.3.4 UC04: Resultat entgegennehmen

Ist die Berechnung auf dem Solver abgeschlossen, wir das Resultat an den Benutzer zurückgesendet. Dieser kann dieses entgegennehmen, herunterladen und darstellen lassen.

1.5.3.5 UC05: Login

Um unerlaubte Zugriffe auf das WebUI zu verhindern, sowie um die verschiedenen Benutzer auseinanderhalten zu können, muss sich der Benutzer beim Start der Applikation einloggen.

2 Analyse

In diesem Kapiteln werden die erforderlichen und wünschenswerten Funktionalitäten der Software analysiert. Dabei wird der Fokus auf die drei Hauptkomponenten der Software gelegt. Dazu gehört ein Silverlight Client, ein Webserver zur Publikation der Silverlight Applikation (zur Vereinfachung nachfolgend nur noch Silverlight Webserver genannt) sowie eine Cloud-Schnittstelle. Hierbei muss angemerkt werden, dass es sich bei der Cloud-Lösung der HSR konkret um einen HPC Cluster, nicht um eine Rechner-Cloud, handelt. In Bezug zur Schnittstelle macht dies jedoch keinen Unterschied. Der Einfachheit halber wird im folgenden Text nur noch von einer Cloud-Schnittstelle für den HPC Cluster oder sonstigem cloud-fähigem Backend gesprochen.

Die Analyse fokussiert sich auf den Simulationsworkflow, der bereits im TransATUI unterstützt wird und auch in der neuen web-basierten Lösung im Wesentlichen implementiert werden soll. Weitere Funktionen, wie zum Beispiel die Projektverwaltung, wurden im bisherigen Workflow nicht angewendet. Sie dienen dazu, das Web Frontend benutzerfreundlicher und lokal unabhängig zu machen.

Für einige Funktionen der oben genannten Hauptkomponenten werden zusätzliche Hilfskomponenten wie zum Beispiel eine Datenbank benötigt. Sowohl alle Funktionen als auch allfällige Zusatzkomponenten werden erklärt und begründet.

Die Analyse erfolgt ohne die Details der Implementation. Auf diese wird im Kapitel 3: Design näher eingegangen. Es ist möglich, dass einige der hier erwähnten Funktionen im Prototyen nicht implementiert werden, sondern lediglich deren Aspekte diskutiert werden.

Weitere Informationen zu TransAT oder TransATUI sowie zum erwähnten Workflow oder den Projekt- und Lösungsdateien, von denen die Rede sein wird, können im Technischen Bericht unter dem „Kapitel 2: Stand der Technik“ gefunden werden.

2.1 Silverlight Client

Der Silverlight Client ist der Hauptteil der Software. Er stellt einer beliebigen Anzahl Benutzern ein grafisches Webinterface zur Verfügung, welches ihnen erlaubt ihre Projekte zu verwalten und zu bearbeiten. Für die Bearbeitung der Projekte können zahlreiche Parameter gesetzt werden. Auch die Eingabe dieser Parameter wird von der grafischen Oberfläche unterstützt. Sobald ein Projekt vollständig bearbeitet wurde, kann dieses an den Solver, der sich in einer Cloud befindet, gesendet werden. Dieser berechnet die eingegebenen Parameter und sendet die erhaltene Lösung an den Client zurück. Darauf nimmt dieser die Lösungsdateien entgegen und ermöglicht es dem Benutzer, die Lösungsdateien herunterzuladen. Der Benutzer kann die Output-Dateien danach mit seinem lokal installierten Visualisierungsprogramm darstellen lassen.

Im Folgenden werden alle Funktionen und Komponenten erklärt, welche für die verschiedenen Aufgaben des Silverlight Clients notwendig sind.

2.1.1 Login

Um auf die Funktionen der Applikation zuzugreifen, muss sich der Benutzer als erstes einloggen. Dies dient gleich zwei verschiedenen Zwecken:

Einerseits können nur zugelassene Personen die Applikation verwenden. Dies ist sehr wichtig, da das Benutzen der Applikation kosten- bzw. lizenzpflichtig ist. Alle zugelassenen Benutzer werden erfasst. Dies kann zum Beispiel per Eintrag in eine Benutzerdatenbank erfolgen. Logt sich ein Benutzer ein, so kann der Silverlight Client per Zugriff auf die Ablage der Benutzerdaten feststellen, ob dieser Benutzer existiert und ob seine Authentifizierung korrekt ist. Ist dies der Fall, so wird dem Benutzer der vollständige Zugriff auf die Applikationsfunktionen erlaubt.

Andererseits wird die Verwaltung der Projekte eines Benutzers und die korrekte Zuweisung von Daten, die zum und vom Solver gesendet werden, vereinfacht, indem die Benutzer eindeutig identifiziert werden können. Es wird ermöglicht, dass die Beanspruchung von Rechenzeit in der Cloud genau auf einzelne Benutzer zugeteilt werden kann und ebenso dass die Lösungsdateien letztendlich wieder an den richtigen Benutzer gelangen können.

2.1.2 Simulationsprojekt

Ein Simulationsprojekt oder kurz Projekt besteht aus mehreren Projektdateien, maximal einem Archiv mit Lösungsdateien, einem Projektnamen, einem Bearbeitungsdatum und es besitzt die Zustände „Editing“, „Ready“, „Solving“ und „Solved“. Der genaue Ablauf ist im Zustandsmodell der Abbildung 5 ersichtlich.

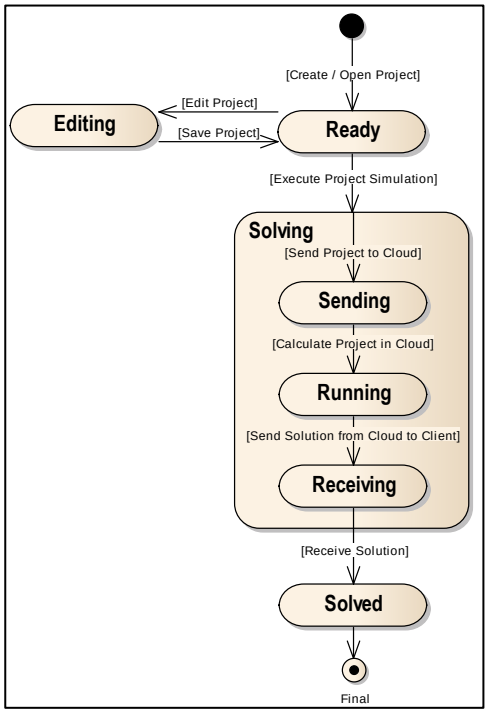

Abbildung 5: Zustandsmodell eines Projekts

Ein neues Projekt befindet sich jeweils im „Ready“, also Bereit. Wird es bearbeitet, so fällt es in den Zustand „Editing“. In diesem Zustand kann das Projekt bearbeitet werden, das heisst Parameter geändert oder sogar ganze Projektdateien hinzugefügt werden. Sobald die Bearbeitung des Projektes abgeschlossen ist, indem es gespeichert wird, wird der Zustand wieder auf „Ready“ gesetzt. Der Wechsel zwischen diesen zwei Zuständen, also „Editing“ und „Ready“, ist möglich, solange sich das Projekt nicht im Berechnen- oder im Abgeschlossen-Zustand befindet.

Ein auf „Ready“ gesetztes Projekt kann zur Auswertung an den Solver gesendet werden. Es erhält einen der Unterzustände vom Zustand „Solving“. Dieser Zustand setzt sich aus drei weiteren Zuständen zusammen. Zuerst wird das Projekt an die Cloud gesendet („Sending“), dann wird es vom Solver berechnet („Running“) und letztendlich wieder an den Client zurückgesendet („Receiving“).

Ist die Lösung beim Client eingetroffen, wird der Status auf „Solved“ gesetzt. Das Projekt ist somit abgeschlossen. Um Inkonsistenzen zwischen dem Input und der Lösung zur vermeiden, kann ein abgeschlossenes Projekt nicht mehr bearbeitet werden. Der Benutzer hat jedoch die Möglichkeit das Projekt mit seinen Inputs aber ohne die Lösungsdatei zu kopieren. Das kopierte Projekt wird in den Zustand „Ready“ gesetzt.

2.1.3 Projektdateien

Die Projektdateien enthalten jeweils ein Teil der Parameter, welche in der Benutzeroberfläche eingegeben werden können. Wird also eine solche Datei eingelesen oder auch ein vollständiges

Projekt (Zusammenstellung mehrerer Projektdateien) geöffnet, so werden die Benutzer-oberflächenparameter, wie zum Beispiel Checkboxen, auf die Zustände abgeglichen, welche in den Projektdateien abgelegt sind.

Lokal abgelegte Projektdateien können einzeln und beliebig in ein Projekt heraufgeladen werden.

Zu den Projektdateien gehören alle Dateien, welche vom Solver für die Berechnung eines Problems benötigt werden.

2.1.4 Lösungsdateien

Sobald der Solver ein Projekt fertig berechnet hat, sendet dieser das Resultat in Form von Lösungsdateien, welche in eine Archiv-Datei (z.B. Zip) zusammengefasst wurden, an den Client zurück. Sobald der Client das Archiv erhält, kann der Benutzer dieses herunterladen.

2.1.5 Projektverwaltung

Alle Projekte, die vom Benutzer erstellt wurden, werden auf dem Silverlight Client verwaltet. Somit ist es möglich, dass Projekte weiterbearbeitet oder dass von einem sehr ähnlichen Projekt einzelne Parameterdateien übernommen werden können. Projekte können erstellt, gelesen, bearbeitet und gelöscht werden.

Dem Benutzer soll auch die Möglichkeit gegeben werden, seine Projekte herunterzuladen um diese lokal auf seinem Rechner zu speichern oder umgekehrt lokale Projekte auf den Silverlight Client heraufzuladen.

Zur Ablage der Projekte ist es von Vorteil, wenn deren Metadaten in einer Datenbank abgelegt werden können. Dabei sollen die Metadaten zusätzlich die Links zu den Files enthalten, welche auf einem Filesystem abgelegt werden.

2.1.6 Projektübersicht

Sobald sich ein Benutzer angemeldet hat, soll ihm eine Übersicht über alle Projekte gezeigt werden. Die Übersicht ermöglicht es auf einen Blick den Status der Projekte zu überprüfen. Dabei wird der jeweilige Zustand des Projekts ersichtlich gemacht. Ein Projekt im Zustand „Solved“ ist in Besitz von Lösungsdateien. Diese können einzeln oder von mehreren Projekten zusammen direkt aus der Übersicht heruntergeladen werden.

2.1.7 Kommunikation mit dem Silverlight Webserver

Wird der Befehl gegeben, ein Projekt zu berechnen, so fasst der Silverlight Client alle Projektdateien des betroffenen Projektes in einer Archiv-Datei zusammen und sendet dieses an den Silverlight Webserver.

Um den Silverlight Client über den Fortschritt der Projektberechnung zu informieren, wird für den Informationsaustausch ein Poll Service beim Webserver erstellt. Das heisst, der Client wir im Abstand von ca. 4 Sekunden eine Fortschrittsabfrage an den Server stellen. Um die einzelnen Projekte auseinanderzuhalten, wird für jede Projektberechnung ein separater Poll-Service gestartet. Durch den einmaligen Benutzernamen sowie dem pro Benutzer einmaligen Projektnamen lassen sich die Projekte unterscheiden. Weiter soll es dem Client auch möglich sein über einen Service Kommandos zu geben, wie zum Beispiel einen Abbruchbefehl.

Erhält der Client über die Fortschrittsabfrage die Meldung, dass die Berechnung komplett ist, weiss er, dass somit auch die Lösung bereit ist. Sobald diese Meldung erfolgt, lädt der Client die in einer Archiv-Datei enthaltene Lösung herunter.

Für die Verwaltung der Projekte und der Benutzerdaten wird eine Datenbank benötigt. Diese befindet sich auf der Seite des Silverlight Webservers. Der Client besitzt somit eine Möglichkeit für den Austausch dieser Daten mit dem Webserver.

2.2 Silverlight Webserver und Cloud-Schnittstelle

2.2.1 Datei- und Nachrichtenübermittlung

Der Webserver besitzt einen Input-Ordner für die Dateien, die an den Solver geschickt werden sowie auch einen Output-Ordner für die Dateien, die von Solver zurückgeschickt werden.

Wenn ein Client ein Projekt in Form einer Archiv-Datei zur Berechnung an den Webserver hoch lädt, wird dieses in dem Input-Ordner abgelegt. Sobald die Datei vollständig ist, wir der Service der Schnittstelle darüber in Kenntnis gesetzt. Dieser sendet die Datei direkt vom Input-Ordner des Webservers zur Berechnung an die Cloud weiter.

Erfolgt eine Poll-Abfrage des Clients, so sendet der Webserver auch diese an den Service der Cloud-Schnittstelle weiter. Dabei hört der Webserver den Informationsaustausch mit. Sobald er feststellt, dass die Berechnung eines Projekts abgeschlossen ist, beauftragt er den Service der Cloud-Schnittstelle die Archiv-Datei mit den Lösungsdateien in den Output-Ordner des Webservers zu legen. Anschliessend sendet er dem Client zusammen mit dem Fortschritt den Pfad auf die Lösungsdatei als Antwort auf die Fortschrittsabfrage.

Die Cloud-Schnittstelle soll über den Webserver Kommandos vom Client erhalten. Sie nimmt diese entgegen und führt das Kommando aus. Ein Kommando kann zum Beispiel den Abbruch oder das Pausieren der Solver-Berechnung bewerkstelligen.

2.2.2 Datenbankverbindung

Der Webserver bietet dem Client Zugriff auf die Benutzer- und auf die Projektverwaltungsdatenbank sowie auf ein Filesystem um alle Projektdateien abzulegen.

3 Design

3.1 Architektonische Darstellung

In der Abbildung 6 ist eine stark vereinfachte Architekturübersicht zu sehen. Die Architektur besteht aus folgenden Hauptkomponenten:

1. Silverlight Client, der im Browser des Benutzers läuft
2. IIS-Webserver, der zwei Funktionen gleichzeitig übernimmt:
    a. Webserver zur Publikation der Silverlight-Applikation und als Service, mit dem der Silverlight Client kommuniziert
    b. Cloud-Schnittstelle als abstrakte einheitliche Schnittstelle zur transparenten Unterstützung verschiedener Cloud Backend-Lösungen (z.B. HPC Cluster oder Cloud Computing Plattform)
3. Eine oder mehrere spezifische Cloud-Backends

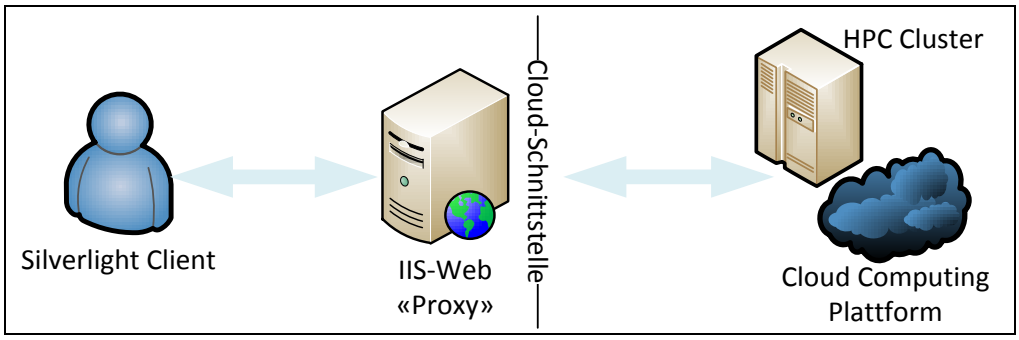

Abbildung 6: Vereinfachte Architekturübersicht

In den folgenden Kapiteln wird auf die Details der einzelnen Komponenten sowie auf die Kommunikation zwischen ihnen eingegangen. Weiter wird auch auf die Architektur der verwendeten Cloud-Backends und deren Anbindung eingegangen.

3.1.1 Silverlight Client

Der Silverlight Client besteht aus folgenden Teilen:

1. Verwaltung der Projekte
2. Umsetzung des Simulations-Workflows im UI
3. Parsen und Generieren der Parameterfiles
4. Upload und Download der Projektdaten
5. Kommunikation mit dem Webserver

Im Folgenden wird auf die Details des Designs dieser Teile eingegangen. Auf UI spezifische Design-Details, wie zum Beispiel die Umsetzung des Simulations-Workflows, wird im Abschnitt UI Design weiter eingegangen.

3.1.1.1 Verwaltung der Projekte

Aus Zeitgründen konnte die geplante Datenbank, welche die Verwaltung der Projekte unterstützen sollte, nicht umgesetzt werden. Aus diesem Grund wurde ein Mock-up der Datenbank erstellt. Beim Design von diesem Mock-up wurde darauf geachtet, dass dieser möglichst einfach durch eine Datenbank ersetzt werden kann. Für das Design wurde das in der Abbildung 7 abgebildete Façade-Pattern angewendet. Die Klasse ProjectDatabase dient dabei als Façade. Sie stellt alle Methoden zur Verfügung, welche für die Verwaltung der Projektdaten benötigt werden. Die ProjectList stellt eine

Liste der Klasse Project dar. Sie enthält die Metadaten zu den Simulationsprojekten, während die Klasse Storage Zugriff auf den isolierten Speicher der Silverlight Applikation gewährt. Somit dient die ProjectList als Mock-up für eine Datenbank, welche die Metadaten persistiert. Die Klasse Storage dient als Mock-up für das Filesystem, welches die einzelnen Projektdateien speichert.

Wird nun der Mock-up durch eine Datenbank und ein Filesystem ausgetauscht, müssen lediglich die ProjectList durch den Datenkontext der Datenbank und der Storage durch das Filesystem ausgewechselt werden. Die Zugriffsmethoden der ProjectDatabase, auf welche die ViewModels und die Services zugreifen, bleiben jedoch dieselben.

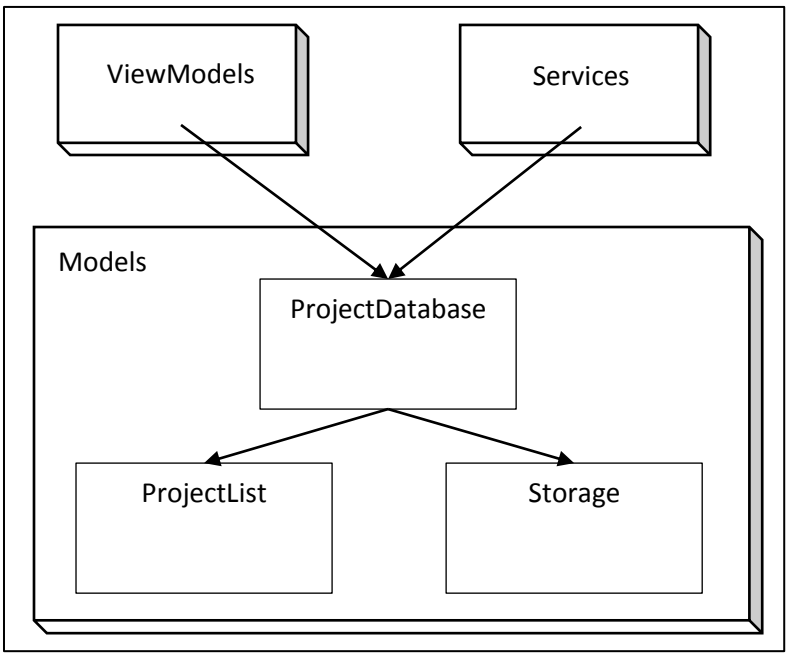

Abbildung 7: Façade Pattern

Die Gründe, wieso diese Lösung nur als Mock-up dient und später durch eine Datenbank und ein Filesystem abgelöst werden sollte, sind folgende:

1. Der isolierte Speicher (Isolated Storage) der Silverlight Applikation befindet sich auf dem Rechner des Benutzers. Da die Dateien eines Simulationsprojektes bis zu einem Gigabyte Platz einnehmen können, ist unter Umständen nicht gewährt, dass der Benutzer genügend freien Speicher auf seiner Festplatte zur Verfügung hat. Das Ziel, die Applikation webbasiert und vom Rechner des Benutzers unabhängig zu halten, ist nicht mehr gewährleistet. Weiter befindet sich der isolierte Speicher in einem Pfad, welcher dem Benutzer nicht bekannt ist. Er könnte diesen versehentlich löschen und somit seine gesamten Projektdaten verlieren.
2. Die ProjectList muss bestehen bleiben, auch wenn der Benutzer die Applikation schliesst. Dies erfolgt, indem beim Beenden der Applikation die Liste serialisiert und im isolierten Speicher abgelegt wird. Beim Öffnen der Applikation wird die Liste wieder deserialisiert und eingelesen. Neben den gleichen Nachteilen, die betreffend dem isolierten Speicher erwähnt wurden, besteht auch das Problem der Serialisierung. Diese verlangsamt den Start und das Beenden der Applikation. Weiter könnten andere Prozesse unerlaubt in den Serialisierungsvorgang eingreifen und die Serialisierung verfälschen, sodass die Liste nicht mehr Deserialisiert werden kann. Die Folgen sind auch hier der Verlust der gesamten Projektverwaltung.

3.1.1.2 Parsen und Generieren der Parameterfiles

Für das Auslesen und Schreiben der Parameter von Projektdateien wurde eine Parser-Klasse geschrieben. Die Funktion des Parsers begrenzt sich dabei auf die Input-Projektdatei eines Simulationsprojektes. Die Input-Projektdatei besteht aus einem ASCII-Text, welcher verschiedene Sektionen, mit den jeweiligen zugehörigen Parametern beinhaltet.

Beim Auslesen der Datei erstellt der Parser pro Sektion ein Dictionary, welcher die einzelnen Parameter der Sektion enthält. Die einzelnen ViewModels, welche die in der Datei enthaltenen Parameter an ihre Views weitergeben, lesen über diese Dictionaries die für sie relevanten Daten aus.

Das Generieren der Datei ist etwas aufwändiger. Der Parser arbeitet auch beim Generieren der Files mit Dictionaries. Die ViewModels schreiben beim Abspeichern eines Projekts die Parameter in die entsprechenden Dictionaries. Danach werden die Dictionaries sektionsweise wieder in die Datei geschrieben. Bei einem Schreibvorgang werden alle Dictionaries neu in die Datei eingefügt. Die bestehenden Einträge werden überschrieben. Es muss also darauf geachtet werden, dass alle ViewModels zuerst ihre Parameter in die Dictionaries eintragen und erst danach die Dictionaries dem Parser übergeben werden.

Für den Schreibvorgang wird das in der Abbildung 8 abgebildete Strategy Pattern eingesetzt. Dabei besitzt jedes ViewModel seine eigene vom Writer abgeleitete Writer-Klasse. Weil jedes ViewModel Parameter von verschiedenen Sektionen des Inputfiles besitzt, wird eine zusätzliche Klasse WriteController eingeführt. Diese enthält jedes Dictionary des Parsers als statisches Attritbut, sowie für jede Writer-Subklasse ein statisches bool-Attribut. Das bool-Attribut wird auf „true“ gesetzt, sobald die entsprechende Klasse ihre Parameter in das Dictionary eingetragen hat.

Bei jedem Write-Vorgang wird überprüft, ob alle bool-Attribute auf „true“ gesetzt sind. Ist dies der Fall, so wird die WriteAllPropertiesToFile()-Methode aufgerufen um die Parameter der Dictionaries im WriterController in die Input-Datei zu schreiben.

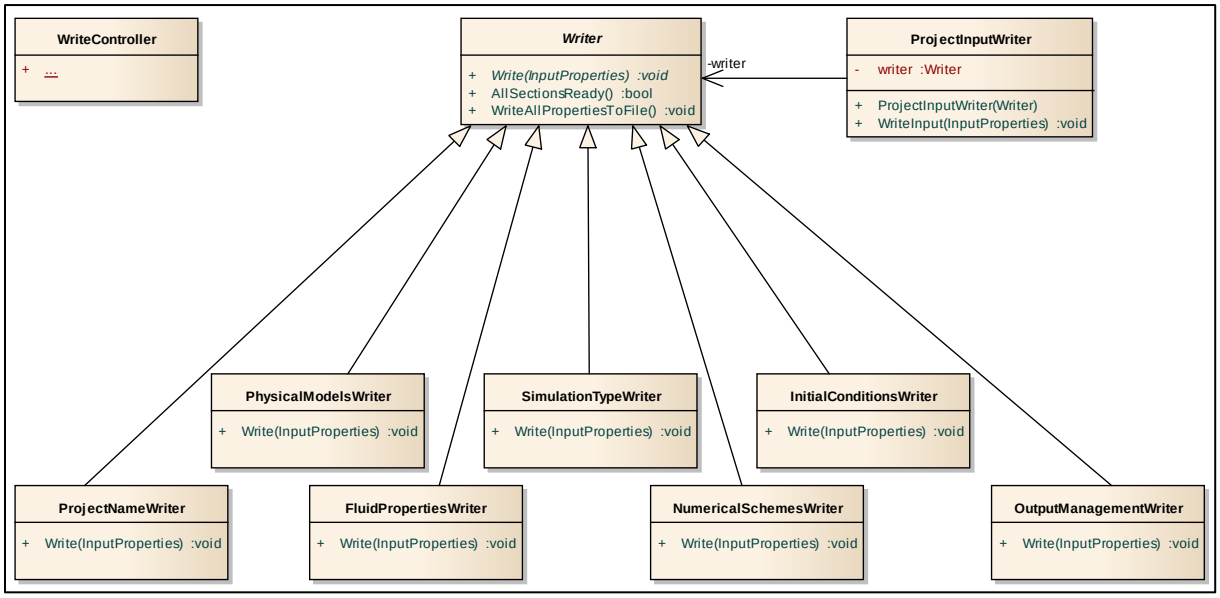

Abbildung 8: Strategy Pattern

3.1.1.3 Upload und Download der Projektdaten

Für den Fileup-/ und Filedownload kommt ein WCF Service zum Zuge. Es werden ausschliesslich Zip-Dateien herauf- oder heruntergeladen. Der Inhalt dieser Dateien wird in einzelne Datenblöcke aufgeteilt und danach blockweise vom Client an den Server oder vom Server an den Client gesendet.

Der Aufruf der Upload-Methode, welche sich auf der Webserver-Seite befindet, erfolgt asynchron vom Client aus. Der asynchrone Aufruf ist vom UI-orientierten Model von Silverlight vorgegeben. Die Datenblöcke werden im Continuation-Style an den Server gesendet. Das heisst, sobald der erste Datenblock erfolgreich gesendet wurde, also ein CompletedEvent vom Server zurückgekommen ist, wird der nächste Datenblock wieder asynchron gesendet. Dies erfolgt solange, bis der letzte Datenblock hochgeladen wurde. Auf der Serverseite wird beim Empfang des ersten Datenblocks eine neue Datei im Ordner, der die Solver-Aufträge enthält, angelegt. Alle folgenden Datenblöcke werde an diese Datei hinzugefügt.

Der Aufruf der Download-Methode erfolgt ebenfalls asynchron in Silverlight von der Seite des Clients aus. Über den Kommunikations-Service, der im Kapitel 3.1.1.4 beschrieben wird, erhält der Client den Pfad der Lösungsdatei sowie die Grösse dieser Datei. Mit diesen Informationen ruft er die Download-Methode auf. Auch hier werden die Daten wieder blockweise gesendet. Aufgrund der bestimmten Dateigrösse ist dem Client auch bekannt, wie viele Datenblöcke er herunterladen muss, bis die Datei vollständig beim Client ist. Die Daten werden Block für Block in ein Objekt der Klasse FileStream eingefügt.

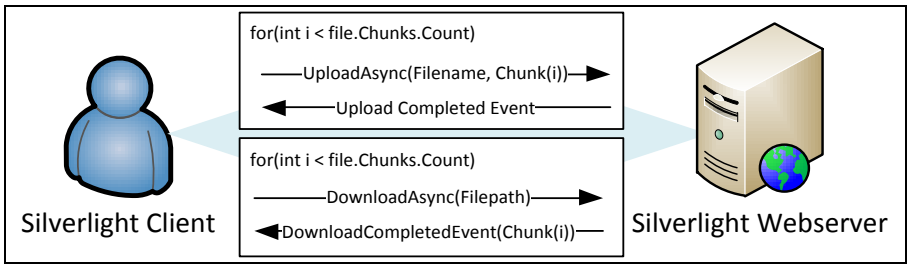

Abbildung 9: Up- und Download mit WCF Service

3.1.1.4 Kommunikation mit dem Webserver

Wird dem Solver ein Auftrag zur Berechnung gesendet, dann kann es beliebig lange dauern, bis der Solver die Berechnung abgeschlossen und die Lösungsdatei zurückgesendet hat. Die Dauer ist aufgrund der Projektgrösse, der angegebenen Simulationsparameter und auch der Auslastung des Netzwerks oder der Cloud Backend sowohl bei der Übertragung als auch bei der Berechnung unterschiedlich lange. Trotzdem muss der Client ohne grosse Verzögerungen die Lösungsdatei erhalten, sobald diese bereit ist. Aus diesem Grund ist pro laufende Berechnung ein Service eingerichtet, der alle 4 Sekunden auf dem Server nachschaut, ob die erwartete Lösung bereit ist. Sobald die Lösung vorhanden ist, gibt der Server über den Callback der Client-Anfrage den Pfad auf die Lösungsdatei sowie deren Dateigrösse bekannt. Dieser Vorgang ist in der Abbildung 10 abgebildet.

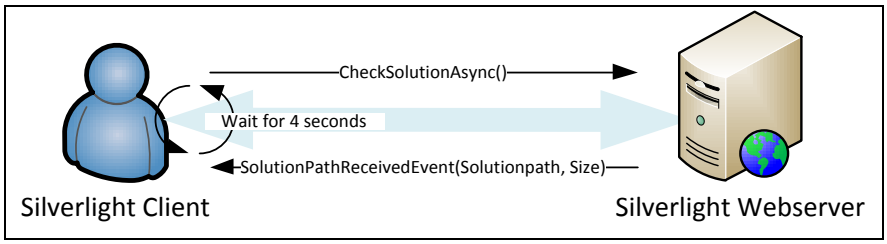

Abbildung 10: Poll-Abfrage vom Client an den Server mit WCF Service

Falls der Client beendet wird während dem eine oder mehrere Berechnungen auf dem Cloud Backend am Laufen sind, wir der Service beim Start der Applikation wieder aufgenommen. Dank dem Status der Projekte stellt er fest, welche Projekte sich beim Beenden der Applikation im Cloud-Backend befunden hatten und eine Lösung erwartet wird.

3.1.2 IIS-Webserver

In der Abbildung 11 ist das Design des Netzwerks dargestellt. Dabei werden die Services in Form von Pfeilen des Weges eines Berechnungsauftrags abgebildet. Es ist ersichtlich, dass der IIS-Webserver den Kern des gesamten Designs darstellt. Er dient dem Hosting der Silverlight Applikation, der Schnittstelle zu den Cloud Backends (HPC und Cloudbroker) und übernimmt auch die Funktion eines Fileshare-Servers. Für das Filesharing werden die Ordner „Job“ und „Solution“ eingesetzt. Sie dienen als Tauschplattform für die Jobs (Berechnungsaufträge) und die Solutions (Lösungsdateien).

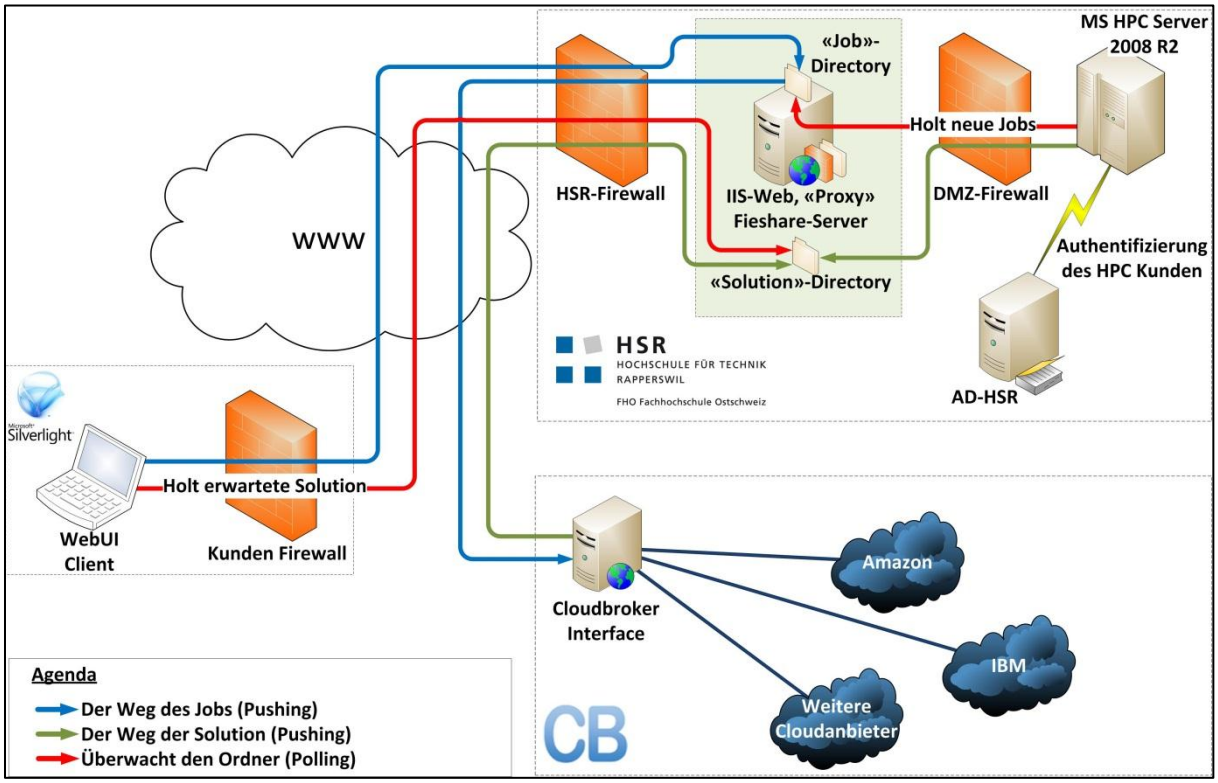

Abbildung 11: Netzwerk Design

3.1.2.1 Silverlight Webserver

Die Aufgaben des Silverlight Webservers sind das Hosting der Silverlight Applikation und die Kommunikation mit dem Silverlight Client und mit der Cloud-Schnittstelle. Er dient sozusagen als Übermittler dieser beiden Komponenten. Die Schnittstelle ist so gelöst, dass der Client seine Service-Aufrufe immer gleicht macht, auch wenn im Hintergrund verschiedene Cloud Backends an die Schnittstelle angeschlossen werden könnten.

File Upload

Bei einem File Upload des Clients nimmt der Webserver eine Datei chunkweise entgegen und speichert diese in dem Job-Directory ab. Der genaue Vorgang ist im Kapitel 3.1.1.3 erklärt. Danach braucht sich der Webserver bei der HPC-Lösung nicht mehr weiter um die Datei zu kümmern, da der HPC das Job-Directory fortlaufend überwacht und komplette Dateien sofort zu sich nimmt. Bei der Cloudbroker Lösung muss der Webserver den Cloudbroker Service, der sich ebenfalls auf dem

Webserver befindet, darüber benachrichtigen, dass eine neue Datei auf dem Job-Directory eingetroffen ist.

File Download

Sobald ein Client beim Webserver anfragt, ob eine Lösungsdatei vorhanden ist, prüft der Webserver in der HPC-Lösung, ob die Lösungsdatei im Solution-Directory liegt. Falls ja, dann leitet er dem Client den Pfad und die Grösse dieser Datei weiter.

In der Cloudbroker-Lösung leitet der Webserver die Anfrage an den HPC weiter. Dieser antwortet mit dem Status des Projekts. Falls dieser Status zeigt, dass die Lösungsdatei bereit ist, beauftragt der Webserver den Cloudbroker Service die Datei in das Solution-Directory zu laden und leitet anschliessend auch hier dem Client den Pfad und die Grösse dieser Datei weiter.

3.1.2.2 Cloud-Schnittstelle

Damit die ganze Architektur möglichst stark von dem verwendeten Cloud-Backend entkoppelt ist, ist eine weitere Schicht dazwischen geschaltet.

Diese Schicht ist in zwei verschieden Varianten implementiert:

1. Die Entkopplung wurde erreicht, indem komplett auf eine softwarebasierte Kommunikation verzichtet und ein Dateisystem verwendet wird. Dies erfolgt über die reine Überwachung der vereinbarten Ordner und über das Verschieben der Dateien in diesen Ordnern.
2. Es wurde ein Webservice geschrieben, der die Minimalanforderung eines cloud-basierten TransAT-Solvers erfüllt. Diese Minimalanforderungen sind folgende:
   - Berechnungsauftrag erstellen
   - Berechnungsauftrag starten
   - Fehlerlog lesen
   - Berechnungs-Log lesen
   - Status erhalten (Erstellt, Läuft, Fertig)
   - Einzelne Ergebnis-Dateien herunterladen
   - Alle Dateien als Packet herunterladen

HPC

Für den HPC bzw. für den HPC-Mock-up wurde die erste Variante der Schnittstellen-Implementierung, also jene über das Dateisystem, gewählt. Der Grund ist, dass sowohl der HPC, als auch sein Mock-up die minimale Anforderung des Webservices aus Variante 2 nicht erfüllen. In der Abbildung 12: Job-Rountrip mit HPC-Server ist der Architektur-Aufbau dieser Lösung abgebildet.

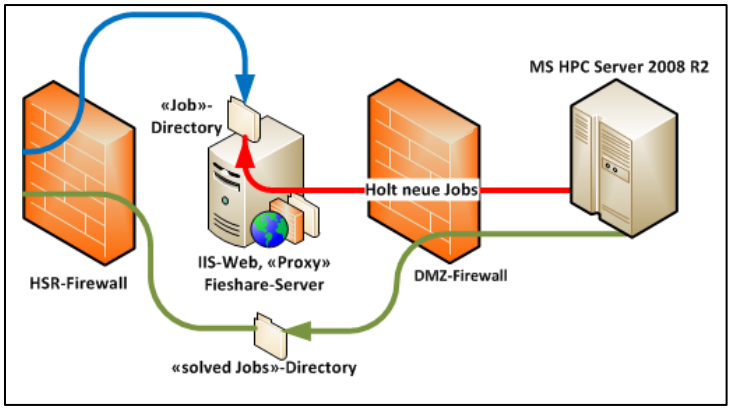

Abbildung 12: Job-Rountrip mit HPC-Server

Wenn die nötigen Vorkehrungen beim HPC der HSR abgeschlossen sind, empfiehlt es sich, die Anbindung an die Silverlight-Applikation über eine Webservice-Schnittstelle zu implementieren und die Lösung über das Dateisystem fallen zu lassen. Dies setzt jedoch voraus, dass der HPC in die DMZ ausgelagert wird oder eine neue DMZ-Firewall-Regel erstellt wird.

Cloudbroker

Die Anbindung an das Cloudbroker-Backend erfolgt über zwei Schichten. Eine schematischte Darstellung davon ist in der Abbildung 13: Cloudbroker Anbindung abgebildet.

Die erste Schicht (Blau) ist der in C# geschrieben Cloudbroker-Client. Im Prinzip ist dieser nichts anderes als eine Erweiterung des vom .Net-Framework zur Verfügung gestellten Web-Clients. Die Erweiterung umfasst alle Funktionalitäten, welche die Cloudbroker-API (Gelb) zum Zeitpunkt des Projektes zu Verfügung stellt (siehe Anhang, Cloudbroker API Manual). Die Zweite Schicht (Grün) bildet den Webservice (CloudbrokerService), welcher den Cloudbroker-Client nutzt um seine Aufgaben zu erfüllen. Der CloudbrokerService wiederum, wird von der Silverlight-Applikation bzw. von deren Webservices (Rot) verwendet.

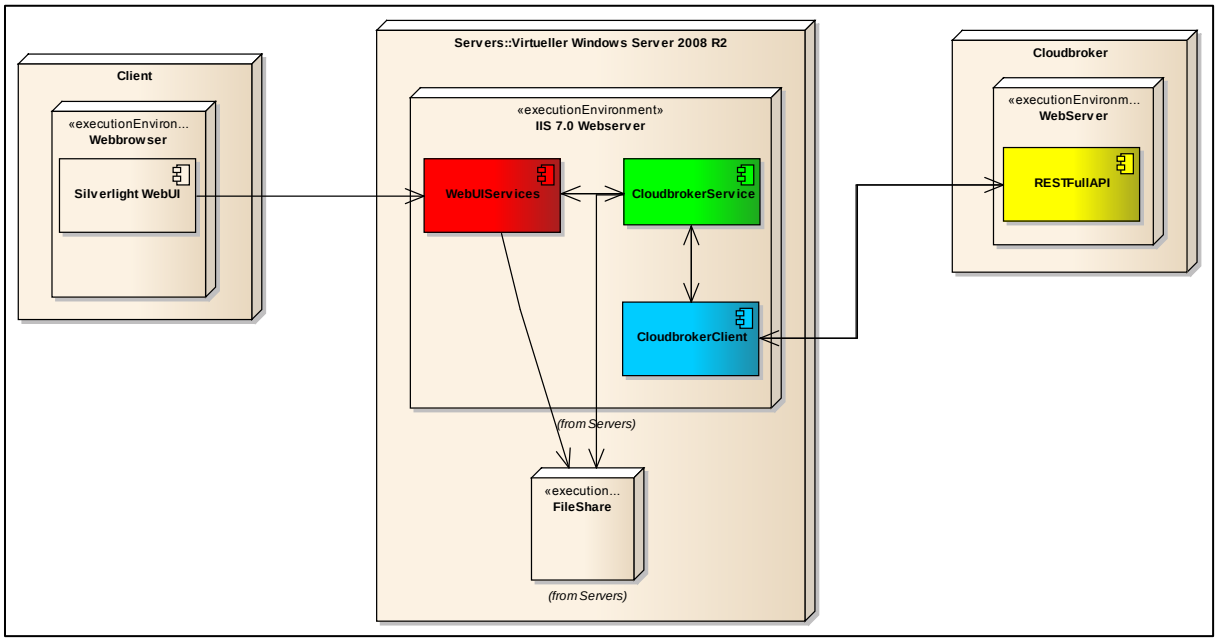

Abbildung 13: Cloudbroker Anbindung

3.2 Assemblies und Namespaces

In den folgenden Abschnitten wird logische Architektur anhand dem Beschrieb der einzelnen Assemblies und Namespaces, aus welchen sich das Softwareprojekt zusammensetzt, beschrieben.

In den beiden untenstehenden Abbildungen ist die Übersicht der Assemblies und Namespaces des Programmcodes abgebildet. Es ist ersichtlich, dass das Architekturdesign der vorherigen Abschnitte auch in der Struktur des Programmcodes wiederspiegelt wird.

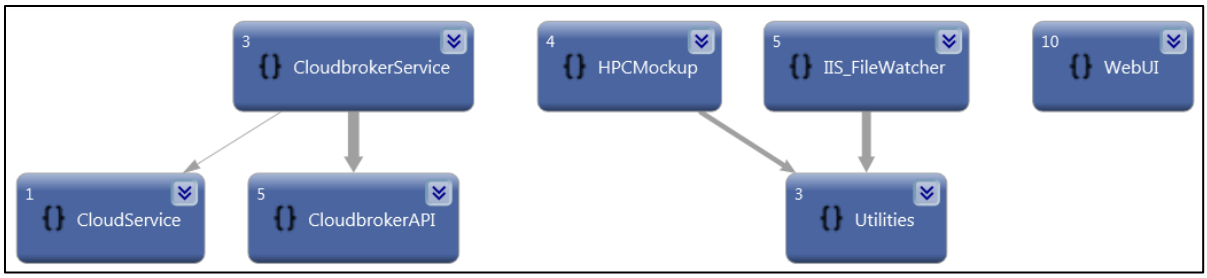

Abbildung 14: Namespace-Übersicht

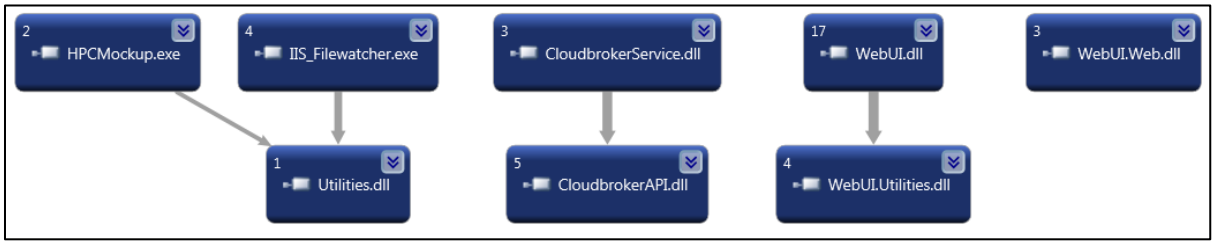

Abbildung 15: Assembly Übersicht

3.2.1 Assembly WebUI

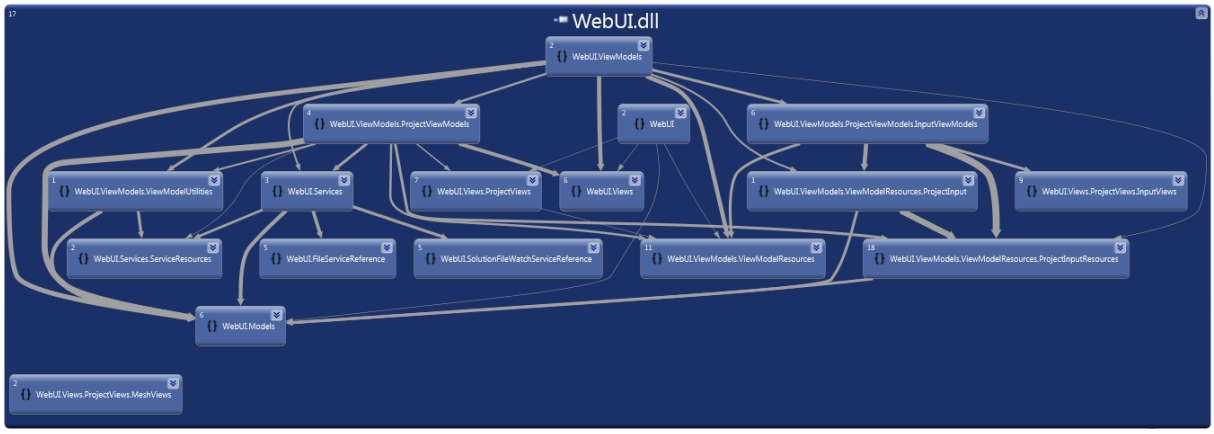

Abbildung 16: WebUI Assembly Übersicht

Der gesamte Code, welcher für die den Silverlight Client benötig wird, befindet sich im WebUI Assembly, welches in der Abbildung 16 dargestellt ist. Das Assembly dient zudem als Startassembly für den Silverlight Client und enthält somit die für die Konfiguration zuständige Datei App.xaml sowie die Startseite MainPage.xaml. Weiter enthält das Assembly die in Tabelle 3.1 ersichtlichen Namespaces:

Namespace	Beschreibung
Views	Dieser Namespace enthält den gesamten Code, der neben der MainPage.xaml-Datei für die Darstellung der Benutzeroberfläche zuständig ist.
ViewModels	Die Daten, welche in den Views dargestellt werden, werden über die ViewModels in diesem Namespace zur Verfügung gestellt. Die ViewModels verhindern, dass die Views direkt auf die im Model enthaltenen Daten zugreifen können.
Models	In diesem Namespace sind die Daten, welche für die Applikation verwendet werden, vorhanden. Sofern die Daten aus einer Datenbank gelesen werden, werden diese über den Webserver hierher gesendet und aufbereitet.
Services	Enthält die Logik, welche für den Zugriff auf die Services des Webservers benötigt wird.

Tabelle 3.1: Namespaces im Assembly WebUI

3.2.2 Assembly WebUI.Web

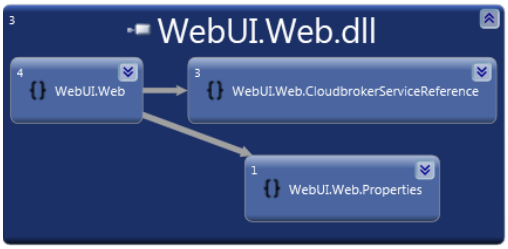

Abbildung 17: WebUI.Web Assembly Übersicht

Das WebUI.Web Assembly stellt die Logik für den Webserver, der die Silverlight-Applikation publiziert, zur Verfügung. Weiter hat er die gesamte Client-Logik in einem xap-Archiv. Dieses Archiv stellt der Webserver jedem Benutzer der auf ihn zugreift zur Verfügung, damit dieser die Silverlight-Applikation starten kann.

Weiter enthält das Assembly die in Tabelle 3.2 ersichtlichen Namespaces:

Namespace	Beschreibung
Web	Enthält die Logik, die der Webserver dem Client zur Verfügung stellt. Der Namespace besteht aus mehreren Webservices. Diese Services ermöglichen dem Client zum Beispiel den Fileup- und Filedownload sowie weitere Kommunikationsfunktionen.
CloudbrokerServiceReference	Dieser Namespace dient als Schnittstelle zum Cloudbroker-Backend.
Properties	Die Konfigurationsdatei befindet sich in diesem Namespace.

Tabelle 3.2: Namespaces im Assembly WebUI.Web

3.2.3 Assembly WebUI.Utilities

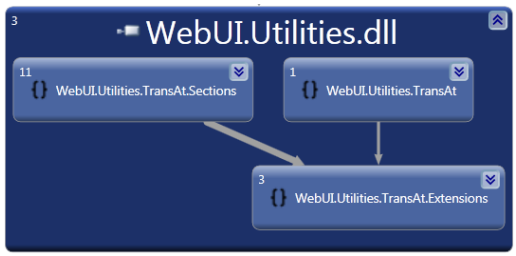

Abbildung 18: WebUI.Utilities Assembly Übersicht

Im Utilities Assembyl befinden sich alle notwendigen Klassen um die Input-Datei „transat.inp“ eines Simulationsprojekts zu lesen und zu schreiben.

Das Assembly enthält die in Tabelle 3.3 ersichtlichen Namespaces:

Namespace	Beschreibung
TransAt	Dieser Namespace beinhaltet den eigentlichen Parser der „transat.inp“-Datei.
TransAT.Sections	Hier sind die einzelnen „Baupläne“ für die einzelnen Sektionen der „transat.inp“-Datei und der TransAtSectionBuilder, welcher die Logik für das Zusammen setzten der einzelnen Sektionen beinhaltet.
TransAT.Extensions	In diesem Namespace befinden sich alle benötigten Extension-Methoden für die Klassen „String“ und „Boolean“ welche für den Parser gebraucht wurden.

Tabelle 3.3 Namespace im Assembly WebUI.Utilites

3.2.4 Assembly HPCMockup

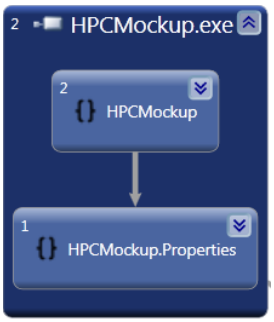

Abbildung 19: HPCMockup Assembly Übersicht

In diesem Assembly befindet sich, wie der Name schon sagt, der HPCMockup. Dabei handelt es sich um ein einfaches, kleines Konsolenprogramm. Es wurde für den Entwicklungsprototyp 2 erstellt, wird jedoch im finalen Prototyp nicht mehr verwendet, da sich der Mock-up des HPC dort erübrigt hat.

Das Assembly enthält die in Tabelle 3.4 ersichtlichen Namespaces:

Namespace	Beschreibung
HPCMockup	Hier befinden sich die HPCMockup Klasse selbst und das eigentliche Programm.
HPCMockup.Properties	Die Konfigurationsdatei befindet sich in diesem Namespace.

Tabelle 3.4 Namespace im Assembly HPCMock-up

3.2.5 Assembly IIS_Filewatcher

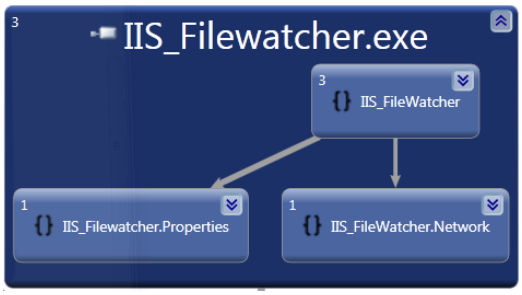

Abbildung 20: IIS_Filewatcher Assembly Übersicht

Auch dieses Assembly wird im finalen Prototyp nicht mehr verwendet. Es wurde aber wie das HPCMockup Assembly im Entwicklungsprototyp 2 verwendet, da es Funktionen für den HPCMockup zur Verfügung stellt.

Das Assembly enthält die in Tabelle 3.5 ersichtlichen Namespaces:

Namespace	Beschreibung
IIS_FileWatcher	In diesem Namespace befindet sich die eigentliche Windowsservice Implementation mit dem entsprechendem Setup-Projekt (Windows Installer).
IIS_FileWatcher.Network	Hier befindet sich die Klasse „NetShare“, welche dafür sorgt, dass der Windowsservice sich über das Netzwerk mit Benutzerinformationen authentifizieren kann und somit die Windowsfreigabe nutzen kann.
IIS_FileWatcher.Properties	Die Konfigurationsdatei befindet sich in diesem Namespace.

Tabelle 3.5 Namespace im Assembly IIS_Filewatcher

3.2.6 Assembly Utilities

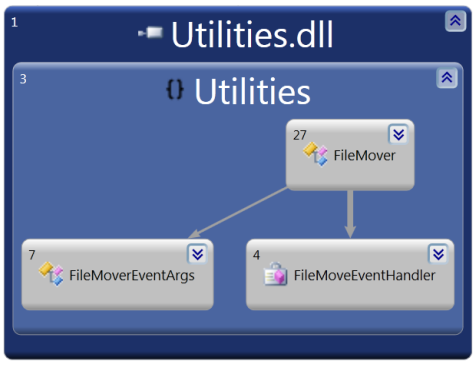

Abbildung 21: Utilities Assembly Übersicht

In diesem Assembly befindet ein einzelner Namespace mit einer Klasse. Die Klasse „FileMover“ wurde zusammen mit ihren Hilfsklassen „FileMoverEventArgs“ und „FileMoveEventHandler“ in dieses Assembly ausgelagert. Der Grund ist, dass die Klasse „FileMover“ sowohl für das Assembly „HPCMockup“ als auch für das Assembly „IIS_Filewatcher“ von zentraler Bedeutung ist. Durch das Auslagern wurde eine maximale Entkopplung erreicht.

3.2.7 Assembly CloudbrokerService

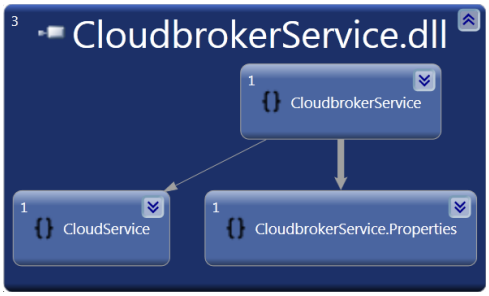

Abbildung 22: CloudbrokerService Assembly Übersicht

Das CloudbrokerService Assembly ist das Assembly des Webservices, welcher die Verbindung zur Cloudbroker-Infrastruktur herstellt.

Das Assembly enthält die in Tabelle 3.6 ersichtlichen Namespaces:

Namespace	Beschreibung
CloudService	In diesem Namespace befindet sich das allgemeine Interface „ICloudService“, welches von jedem Webservice implementiert werden soll, der eine Cloud-Infrastruktur zu Verfügung stellen will.
CloudbrokerService	Hier befindet sich die gleichnamige Klasse „CloudbrokerService“, welche den Service bereitstellt.
CloudbrokerService.Properties	Die Konfigurationsdatei für den Webservice „CloudbrokerService“ befindet sich in diesem Namespace.

Tabelle 3.6 Namespace im Assembly CloudbrokerService

3.2.8 Assembly CloudbrokerAPI

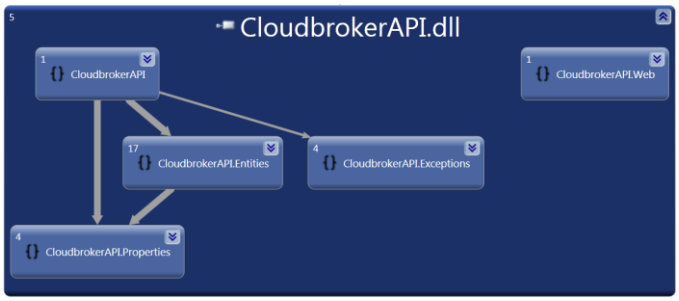

Abbildung 23: CloudbrokerAPI Assembly Übersicht

Das CloudbrokerAPI Assembly macht die RESTFull API der Cloudbroker-Infrastruktur zugänglich. Das Assembly kümmert sich automatisch um die Authentifizierungen bei Cloudbroker, um die Problematik mit den SSL-Zertifikaten und um die De-/Serialisierung von Objekten zu XML-String.

Das Assembly enthält die in Tabelle 3.7 ersichtlichen Namespaces:

Namespace	Beschreibung
CloudbrokerAPI	In diesem Namespace befindet sich der eigentliche CloudbrokerClient, welcher alle Anfragen an die RESTFullAPI weiterleitet sowie die Authentifikation und das SLL-Zertifikat-Management übernimmt.
CloudbrokerAPI.Entities	In diesem Namespace sind alle Klassen die zur De-/Serialisierung benötig werden. Also im Prinzip ein XML-Attribut, Property mapping.
CloudbrokerAPI.Exceptions	Hier befinden sich alle Exceptions, die beim Umgang mit dem CloudbrokerClient auftreten können. Dazu gehören z.B. „NoJobIdException“, „ErrorByDeletingException“ und weitere.
CloudbrokerAPI.Web	Hier ist eine einfache Enum-Klasse abgelegt, welche die diversen Http-Header-Methoden wie POST, GET, DELETE, usw. beinhaltet.
Cloudbroker.Properties	Hier werden in diversen Settings-Dateien die Standartwerte für einen Job gesetzt oder die Login-Informationen geändert.

Tabelle 3.7: Namespace im Assembly Cloudbroker API

3.3 UI Design

Das Design der Benutzeroberfläche ist so aufgebaut, dass es einfach zu bedienen ist und dass der Benutzer in seinem Arbeits-Prozess unterstützt wird. Die Oberfläche ist selbsterklärend und lässt eine Bedienung ohne vorheriges Studieren einer Benutzeranleitung zu. Es wird jedoch vorausgesetzt, dass der Benutzer über die erforderlichen physikalischen Kenntnisse der Fluid Dynamik verfügt.

In den folgenden Abschnitten werden die einzelnen Elemente und Funktionen des UIs aufgezeigt. Weiter wird mit kleinen Szenarien die Benutzerfreundlichkeit demonstriert.

3.3.1 Projektverwaltung

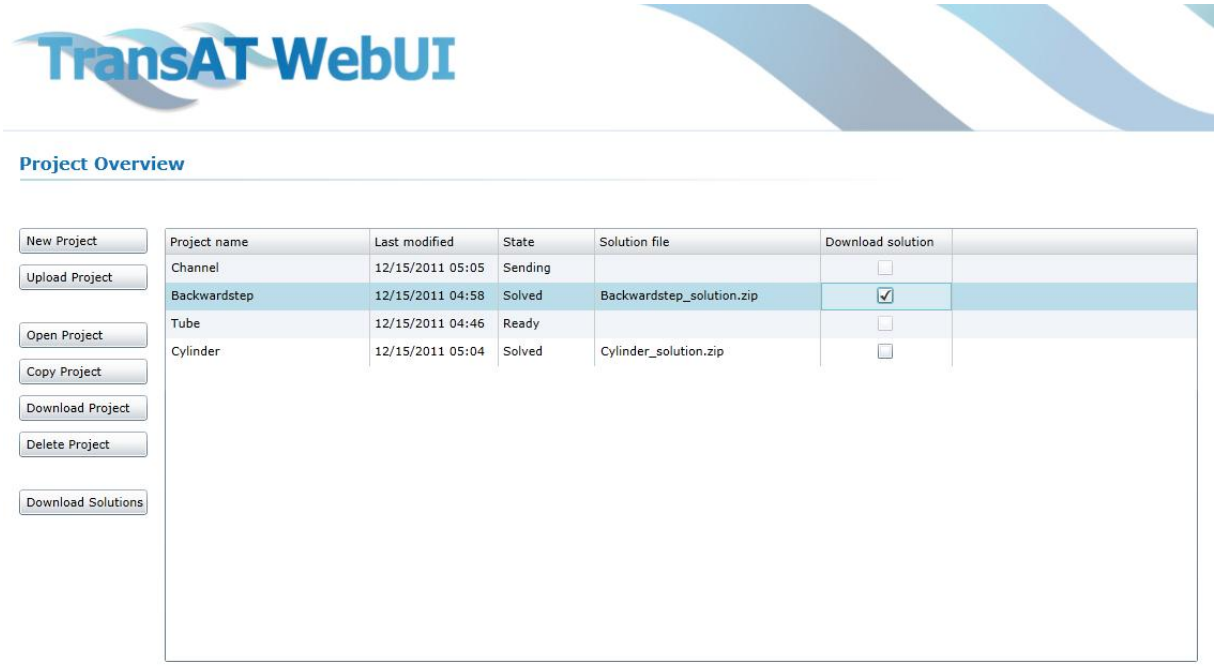

Abbildung 24: UI Projektverwaltung

Beim Start der Applikation wird als Einstiegsseite eine Übersicht über die bestehenden Simulationsprojekte angezeigt. Der Benutzer erkennt auf einen Blick den Status der Projekte. Weiter kann er mehrere Lösungsdateien, die ebenfalls in der Übersicht angezeigt werden, gleichzeitig herunterladen und diese lokal mit einem Visualisierungsprogramm öffnen und darstellen lassen.

Szenario: *Der Benutzer erstellt an einem Arbeitstag die vier Projekte „Channel“, „Backwardstep“, „Tube“ und „Cylinder“. Drei Projekte davon stellt er fertig und sendet diese vor dem Feierabend zur Berechnung los. Am nächsten Morgen startet er die Applikation und stellt auf einen Blick fest, dass über Nacht zwei Projekte fertig berechnet wurden, das dritte jedoch noch in Berechnung ist. Er möchte nun die Lösungen herunterladen und darstellen lassen. Er markiert beide Lösungen für den Download, betätigt den Button für den Download, legt den Speicherort fest und lädt die Dateien somit herunter.*

Für die Verwaltung der Projekte gibt es verschiedene Funktionen. Dazu gehören „New Project“ und „Upload Project“, welche jederzeit wählbar sind. Die Funktionen „Open Project“, „Copy Project“, „Download Project“ und „Delete Project“ jedoch sind nur dann wählbar, wenn ein Projekt in der Liste angewählt wurde. In der Abbildung 25 ist ersichtlich, dass im Moment kein Projekt angewählt wurde und die Button der projektspezifischen Funktionen somit ausgegraut sind.

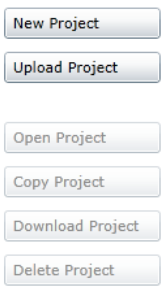

Abbildung 25: UI: Verwaltungsfunktionen

3.3.1.1 Restriktionen und Fehlermeldungen

Bei der Selektion von Lösungen für den Download sind nur Projekte anwählbar, die auch eine Lösung besitzen. Bei den Anderen sind die Checkboxen ausgegraut.

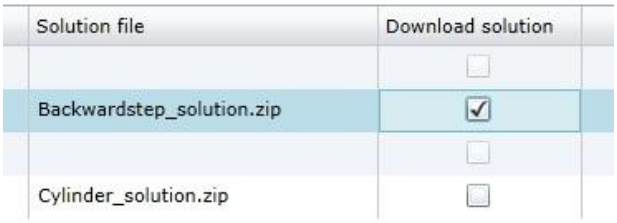

Abbildung 26: Solution Checkboxen

Projekte, die zur Berechnung an den Solver gesendet wurden, können nicht gelöscht werden, solange deren Lösung noch nicht eingetroffen ist. Wird der Befehl trotzdem ausgeführt, erscheint die Fehlermeldung der Abbildung 27. Wenn der Befehl zum Download von Lösungen gegeben wird, jedoch keine Lösung selektiert ist erfolgt die Fehlermeldung der Abbildung 28. Weitere Restriktionen wurden beim Projektupload eingeführt. Einerseits ist es nur möglich Dateien mit TransAT-Dateiendungen hochzuladen. Weiter wird die Input-Datei „transat.inp“ zwingend benötigt. Ist diese in den, für den Projektupload ausgewählten Dateien, nicht vorhanden, dann wird die Fehlermeldung der Abbildung 29 angezeigt. Ist die Input-Datei zwar vorhanden, jedoch fehlerhaft, kommt die Fehlermeldung der Abbildung 30 zum Zuge. Die Datei wird als fehlerhaft erkannt, wenn der Projektname nicht ausgelesen werden kann.

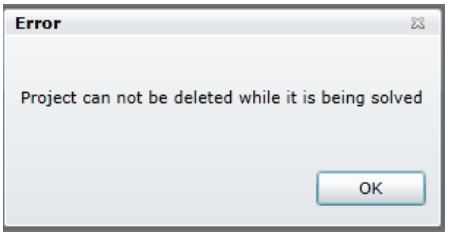

Abbildung 27: UI „Delete“ Error

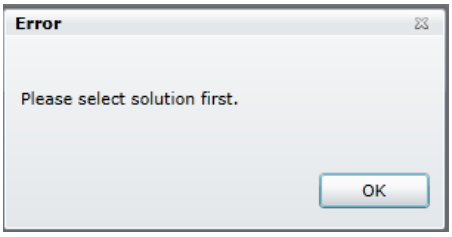

Abbildung 28: UI "No Solution Selected" Error

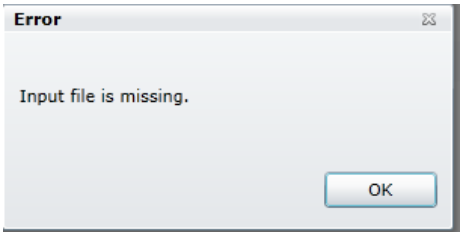

Abbildung 29: Inputfile fehlend bei Upload Error

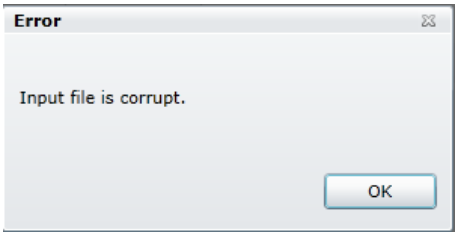

Abbildung 30: Inputfile fehlerhaft Upload Error

Wird beim Erstellen oder beim Kopieren eines Projekts ein bereits vorhandener Projektname eingegeben oder das Eingabefeld leer gelassen, dann wird dies verhindert und mit der passenden Fehlermeldung signalisiert. Das „New Project“- und das „Copy Project“-Fenster verhalten sich beide gleich.

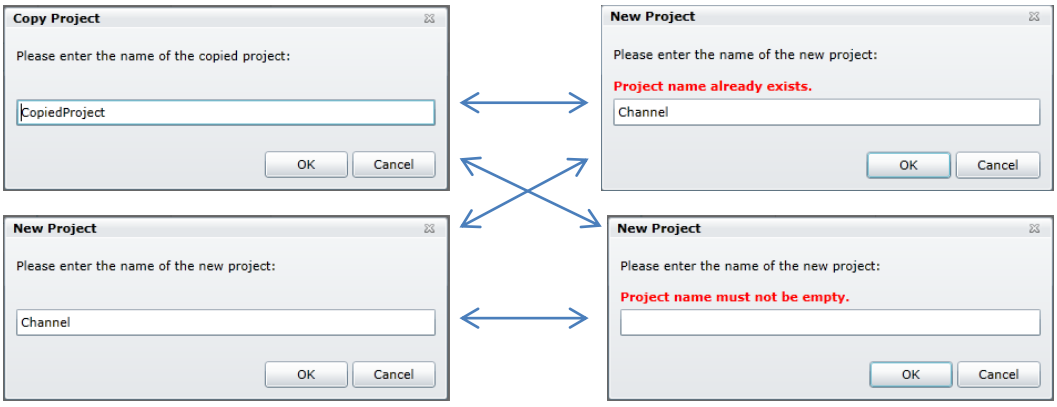

Abbildung 31: Kopieren und Öffnen eines Projekts

3.3.2 Projekt bearbeiten

Wir ein Projekt geöffnet oder ein neues Projekt erstellt, dann wird die Projektbearbeitung geöffnet. Das aktuell geöffnete Projekt wird jeweils als an der Stelle angezeigt, die in der Abbildung 32 rot markiert ist. Es kann also jederzeit festgestellt werden, welches Projekt gerade geöffnet ist, denn während der Bearbeitung ändert sich nur die grün markierte Fläche, der Rest ist statisch.

Die Buttons der Funktionen „Save Project“ und „Close Project“ sind ebenfalls ständig sichtbar. Dies ist sinnvoll, da es dem Benutzer jederzeit möglich sein soll, das Projekt zu Speichern oder zu Schliessen. Wenn der „Save Project“ Button geklickt und somit das Projekt gespeichert wurde, bleibt der Button ausgegraut, bis wieder eine Änderung am Projekt gemacht wird.

Abbildung 32: Startansicht der Projektbearbeitung

3.3.2.1 Projektfiles verwalten

In der „Project“-Ansicht werden die wesentlichen Teile des Projekts, der Projektname und die Projektdateien, bearbeitet und verwaltet. Der Inhalt dieser Projektdateien kann entweder per Input in den folgenden „Mesh“- und „Input“-Fenstern eingegeben werden, direkt über den Editor, welcher

in der Abbildung 34 ersichtlich ist, bearbeitet werden oder sie können von einem lokalen Speicherort des Benutzers direkt in das geöffnete Projekt hochgeladen werden.

Szenario: *Ein Benutzer möchte nur eine kleine Änderung an einer Datei durchführen und weiss genau wie er diese auf einfache Weise direkt in der Datei vornehmen kann. Er öffnet den File Editor, schreibt einige wenige Zeilen Input, speichert die Datei und schliesst den Editor. Er erspart sich somit die Zeit, sich zuerst an die richtige Parameter-Eingabestelle im UI durchzuklicken.*

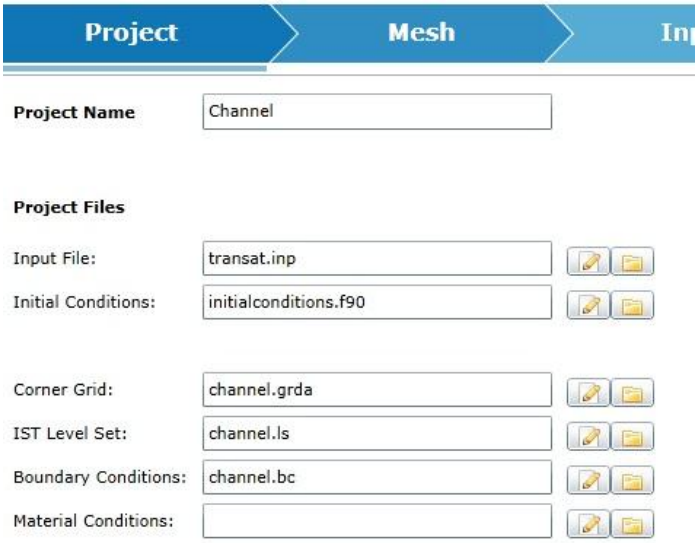

Abbildung 33: „Project“ Ansicht

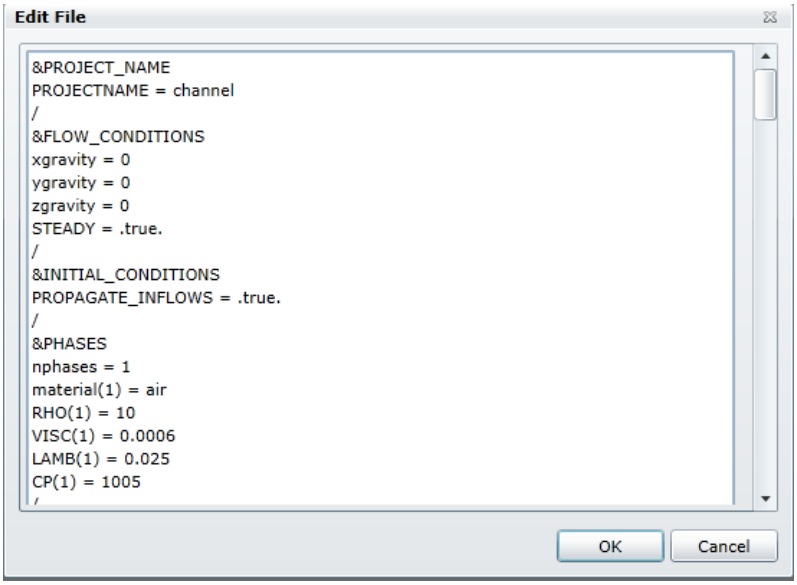

Abbildung 34: File Editing

3.3.2.2 Mesh Parameter

Die Parametereingabe der „Mesh“ Ansicht wurde in dem UI für den Prototyp nicht implementiert. Da jedoch das „Mesh“ ein wesentlicher Teil des Workflows ist, wurde eine nicht funktionale und deshalb ausgegraute Beispielansicht erstellt. Die Parameter, welche hier eingetragen werden könnten, werden im Prototyp durch das Hochladen der entsprechenden Dateien oder durch die Bearbeitung der Dateien über den Editor umgangen.

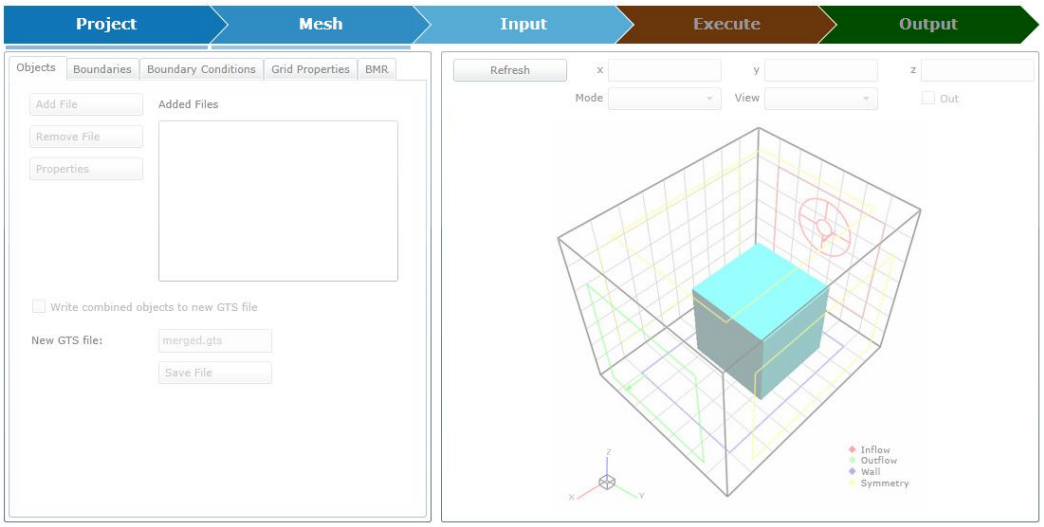

Abbildung 35: Muster der „Mesh“ Ansicht

3.3.2.3 Input Parameter

In der „Input“ Ansicht werden die Parameter der Input-Datei „transat.inp“ eingetragen. Bei jedem Speichervorgang werden die eingegebenen Parameter in die Datei geschrieben. Die Inputs sind verschiedenen Haupt- und Unterkategorien zugeordnet. Die Hauptkategorien werden mittels Tabs unterteilt, während sich die Unterkategorien zwar in der gleichen Ansicht befinden, jedoch visuell durch Trennlinien voneinander getrennt sind.

In der Abbildung 38 und in der Abbildung 42 sind zusätzliche Buttons auf der Ansicht vorhanden. Mit diesen werden „Child Windows“, also kleine Fenster, geöffnet, welche erweiterte Parametereingaben zulassen. Die „Child Windows“ sind in der Abbildung 39: „Adaptive Time Stepping”, der Abbildung 43: „Plane 2D Output“ und der Abbildung 44: „Output Variables“ ersichtlich.

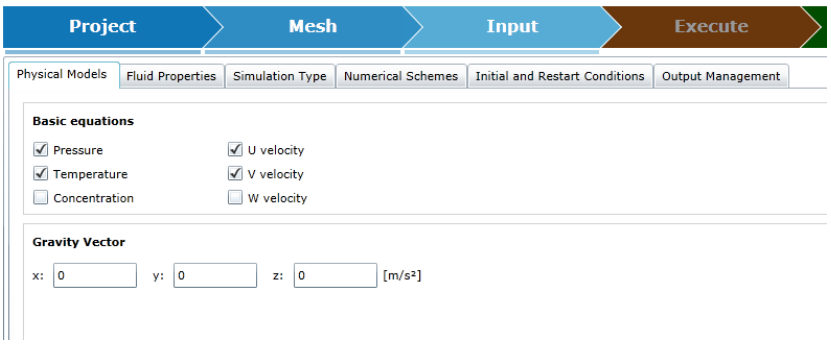

Abbildung 36: Ansicht der „Physical Models“ Input-Parameter

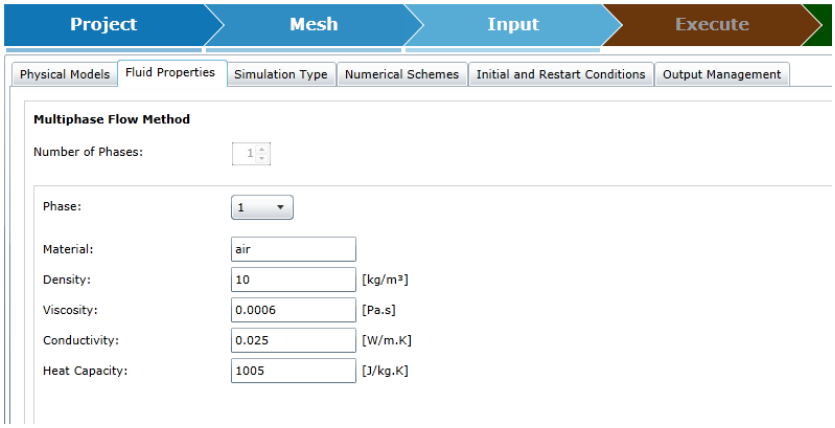

Abbildung 37: Ansicht der „Fluid Properties“ Input-Parameter

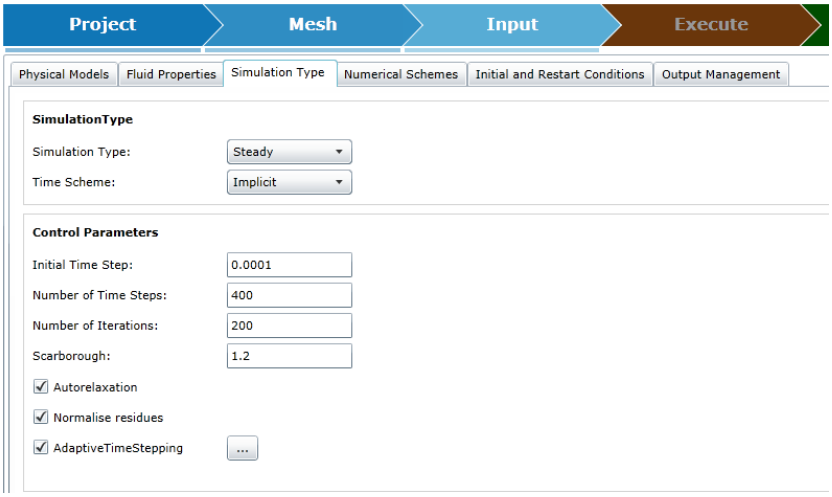

Abbildung 38: Ansicht der „Simulation Type“ Input-Parameter

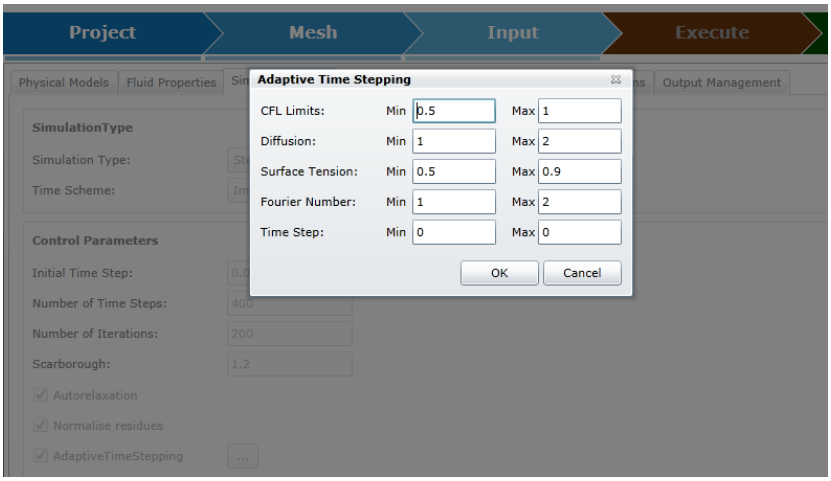

Abbildung 39: „Adaptive Time Stepping”

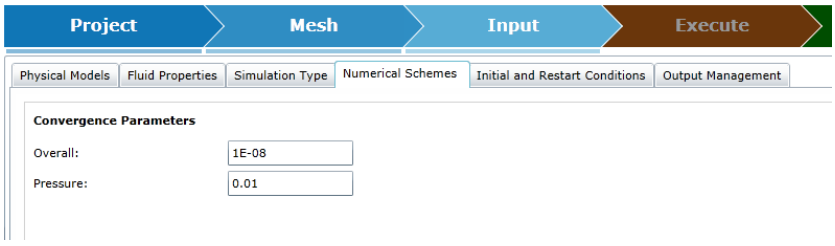

Abbildung 40: Ansicht der „Numerical Schemes“ Input-Parameter

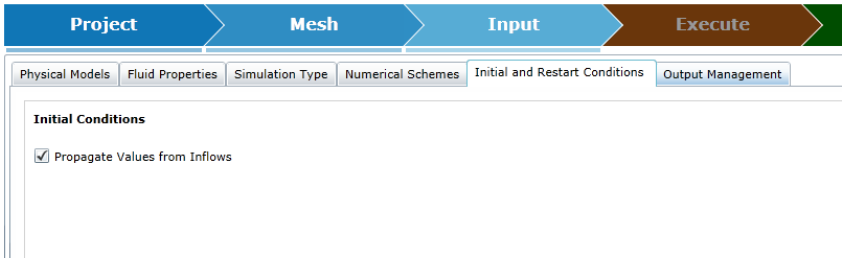

Abbildung 41: Ansicht der „Initial and Restart Conditions“ Input-Parameter

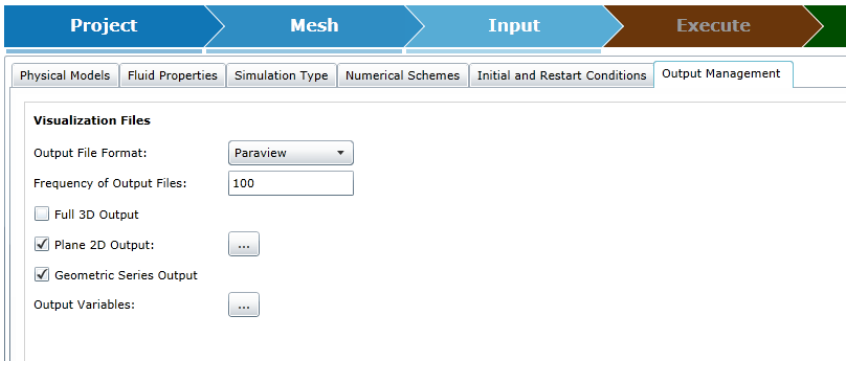

Abbildung 42: Ansicht der „Output Management“ Input-Parameter

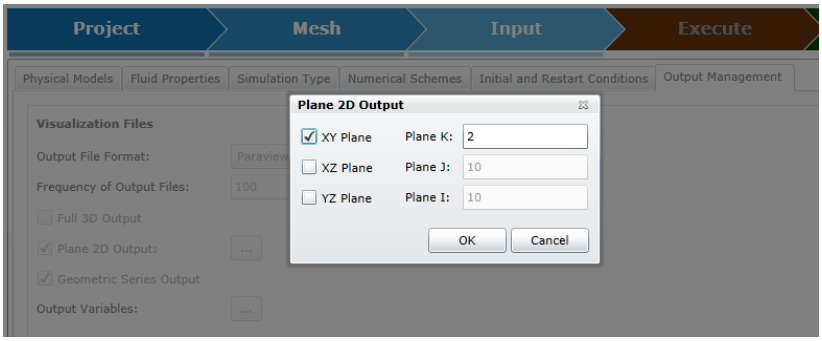

Abbildung 43: „Plane 2D Output“

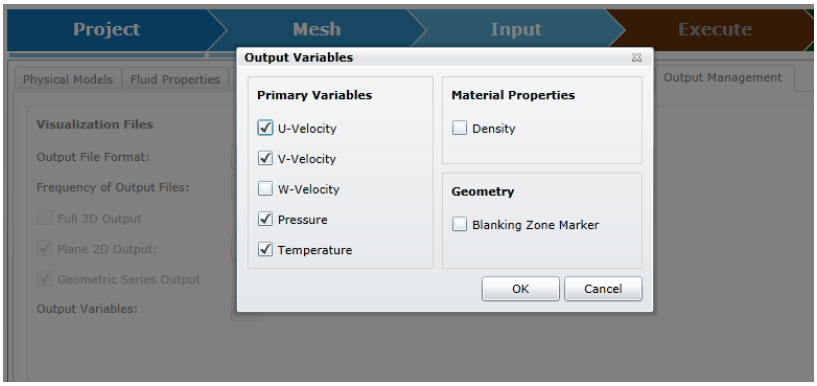

Abbildung 44: „Output Variables“

3.3.2.4 Execute

Sobald die Eingaben im Projekt gespeichert sind, wird die „Execute“ Ansicht aktiv und kann geöffnet werden. Diese Ansicht besitzt lediglich zwei Funktionen: Zum einen kann der Befehl „Solve Project“ gegeben werden um das Projekt im Cloud-Backend berechnen zu lassen. Zum anderen kann das „Execute Log“, das den Status des Projekts auflistet, gelöscht werden. Wenn der „Solve Project“ Button einmal geklickt wurde, dann wird dieser inaktiv, um zu verhindern, dass das Projekt versehentlich mehrere Male an den Solver gesendet wird.

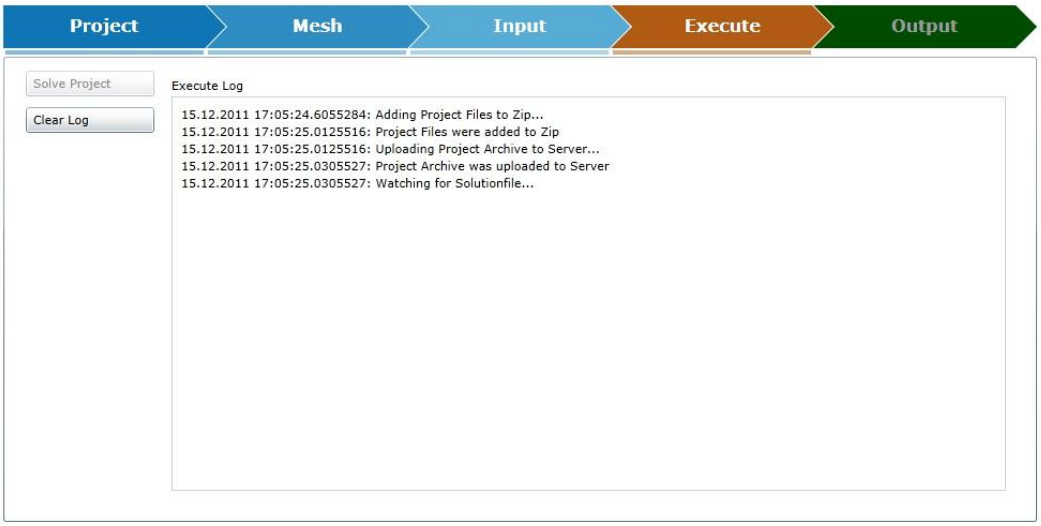

Abbildung 45: „Execute“ Ansicht

3.3.2.5 Output

In der „Output“ Ansicht lässt sich die berechnete Lösungsdatei herunterladen. In einer erweiterten Version des UIs würde hier auch das Log der Berechnung angezeigt werden und statt einzig dem gesamten Zip-Archiv könnten auch einzelne Lösungsdateien heruntergeladen werden.

Abbildung 46: „Output“ Ansicht

3.3.2.6 Workflow

Der Workflow oder in anderen Worten der Arbeitsfluss, den der Benutzer bei jedem Erstellen einer Projektsimulation durchgehen muss, wird unterstützt, indem in der Ansicht der Projektbearbeitung sogenannte Breadcrumbs eingesetzt werden. Die Breadcrumbs zeigen dem Benutzer den Verlauf des Arbeitsflusses, die aktuelle Position und die wählbaren Positionen an. Für die Anzeige der aktuellen Position befindet sich unterhalb des aktuellen und der vorhergehenden Breadcrumbs ein kleiner Fortschrittsbalken. Nicht zugängliche Breadcrumbs sind mit einem Schatten überdeckt.

Abbildung 47: „Project“-Breadcrumb aktiv

Abbildung 48: „Mesh“-Breadcrumb aktiv

Abbildung 49: „Input“-Breadcrumb aktiv

Abbildung 50: „Execute“-Breadcrumb aktiv

Abbildung 51: „Output“-Breadcrumb aktiv

3.3.2.7 Restriktionen und Fehlermeldungen

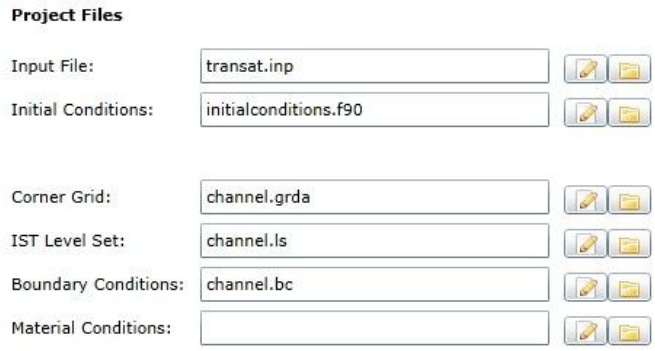

Abbildung 52: Restriktionen bei der Projektdatei-Verwaltung

Bei dem Upload der einzelnen Projektdateien können nur Dateien mit den passenden Dateiendungen hochgeladen werden. Weiter wird beim Umbenennen eines Dateinamens überprüft, ob die Dateiendung noch vorhanden ist. Fall nicht, dann wird diese automatisch wieder hinzugefügt.

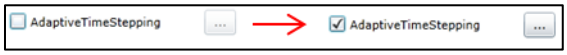

Abbildung 53: Restriktion bei Abhängigkeit

Einige Funktionen erwarten die Erfüllung einer Bedingung bevor sie ausgeführt werden können. So muss zum Beispiel in der Demonstration der Abbildung 53 zuerst das „AdaptivTimeStepping“ aktiviert werden, bevor derjenige Button aktiv wird, der das Fenster mit den zusätzlichen „AdaptivTimeStepping“-Parameter öffnet.

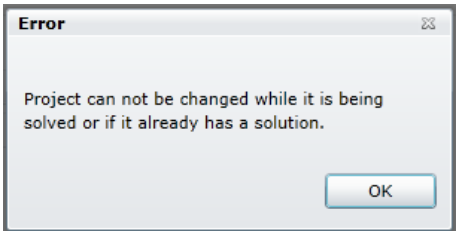

Abbildung 54: Bearbeiten Error

Um Inkonsistenzen zwischen den Projektsimulations-Parameter und der zugehörigen Lösung zu verhindern, kann ein Projekt nicht mehr bearbeitet werden, wenn es einmal an den Solver gesendet wurde. Falls das Projekt trotzdem bearbeitet werden soll, dann hat dem Benutzer die Möglichkeit, das Projekt in der Projektverwaltung zu kopieren und unter einem neuen Namen abzuspeichern. Die Kopie kann dann bearbeitet werden.

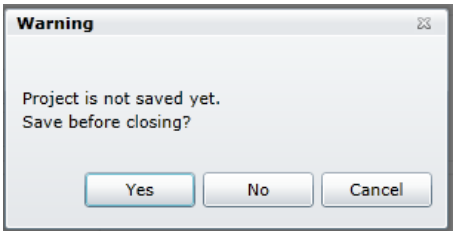

Abbildung 55: Nicht gespeichert Warnung

Wird ein Projekt geschlossen ehe es gespeichert wurde, dann wird der Benutzer darauf hingewiesen. Er hat die Möglichkeiten, das Projekt vor dem Schliessen automatisch speichern zu lassen oder ohne zu Speichern das Projekt zu verlassen sowie zum geöffneten Projekt zurückzukehren.

3.3.3 Design Stil

3.3.3.1 Farblich

Die farbliche Gestaltung des UIs wurde in möglichst dezenten Farben gehalten. Grundsätzlich wurde, orientiert an dem Logo von TransAT, mit verschiedenen Blautönen gearbeitet. Einzig die Farben des „Execute“- und des „Output“-Breadcrumbs weichen von den Blautönen ab. Für „Execute“ wurde eine rötliche Farbe gewählt, damit der Benutzer gewarnt ist, dass es sich hierbei um eine Funktion handelt, die nicht ohne Bedacht verwendet werden soll. Es sollten nämlich nur Projekte abgesendet werden, die bestimmt abgeschlossen sind, da der Berechnungsbefehl mit Kosten verbunden ist. Der „Output“ wurde grün gewählt, hiermit grünes Licht für den Download der Lösung gegeben wird. Es wurde darauf geachtet, dass trotzt der Verwendung von Farben, das UI auch mit einer Sehschwäche immer noch gut zu bedienen ist. In der Abbildung 57 lässt sich feststellen, dass die einzelnen UI Elemente trotzt fehlender Farbe immer noch gut zu unterscheiden sind.

3.3.3.2 Stilistisch

Beim Stil wurde vor allem auf eine einheitliche Gestaltung des UIs Wert gesetzt. So befinden sich z.B. alle Menu Buttons am linken Rand und sie haben alle die gleiche Grösse. Die Elemente werden über die Ansichten hinweg möglichst gleich angelegt, dass selbst bei einem Ansichts-Wechsel keine Unruhe in der Darstellung aufkommt.

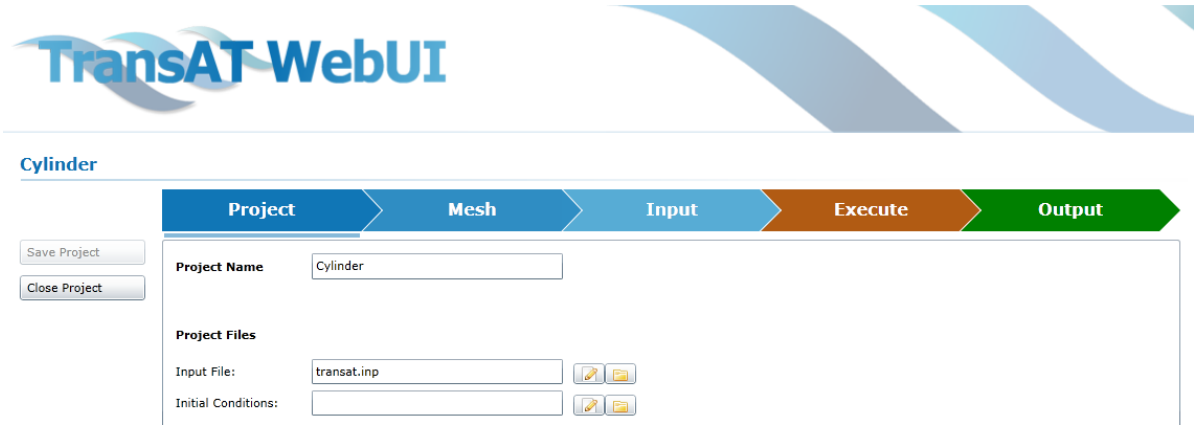

Abbildung 56: UI Design

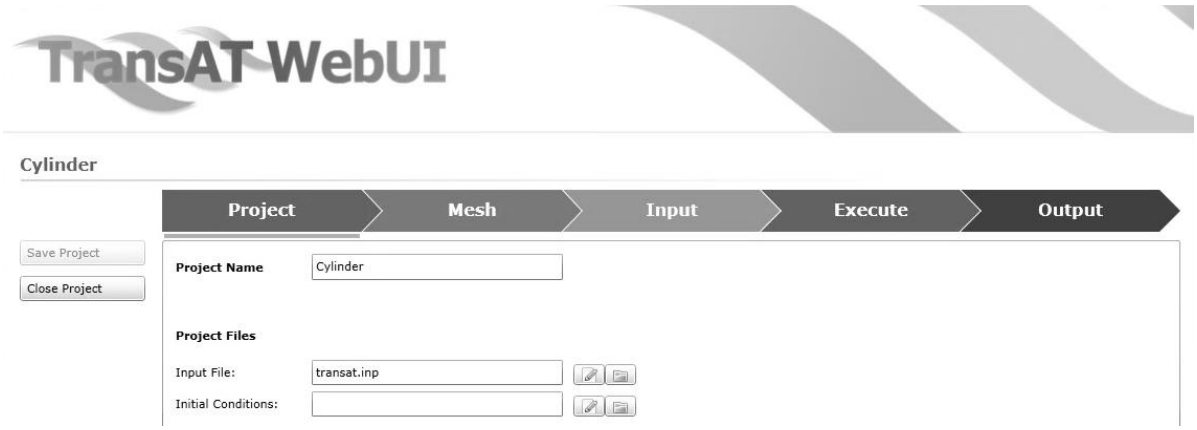

Abbildung 57: UI Design mit fehlender Farbunterscheidung

3.3.4 Implementation spezieller UI Elemente

3.3.4.1 Message Window

Für die verschiedenen Nachrichtfenster mit Hinweisen, Warnungen und Fehlern wurde eine Vorlage erstellt. Dieser Vorlage werden jeweils dynamisch der Fenstertitel, der Anzeigetext und die Auswahl an Knöpfen zugewiesen. Wenn der Benutzer auf dem „Message Window“ einen Button klickt, wird im Hintergrund ein „Event“ ausgelöst. Die aufrufende Klasse nimmt den „Event“ über einen „EventHandler“ entgegen und kann das Resultat des „Message Windows“ abrufen. Das Resultat zeigt an, welchen Button der Benutzer geklickt hat und erlaubt der aufrufenden Klasse die aufgrund des Resultats erforderlichen Funktionen auszuführen.

Das „Message Window“ ist eine Anlehnung an die bereits bestehende WCF- und Silverlight-„MessageBox“. Diese wäre zwar noch einfacher einzusetzen, jedoch ist sie in Silverlight nur in einer abgespeckten Version vorhanden. Das heisst, sie kann nur den „OK“ und / oder den „Cancel“ Button anzeigen. Zudem verwendet sie nicht das Silverlight-Design, sondern das normale WCF-/ Windows-Design.

3.3.4.2 Busy Indicator

Bei länger dauernden Funktionen wird der Busy Indicator dargestellt. Dieser ist ein Element aus dem Silverlight Toolkit. Da der Busy Indicator einen zusätzlichen Thread für die länger dauernde Funktion benötigt, wird die Funktionsweise des Busy Indicators in dem folgenden Abschnitt 3.4: Threads erklärt.

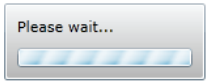

Abbildung 58: Busy Indicator

3.4 Threads

3.4.1 SolutionFileWatcher Backgroundworker

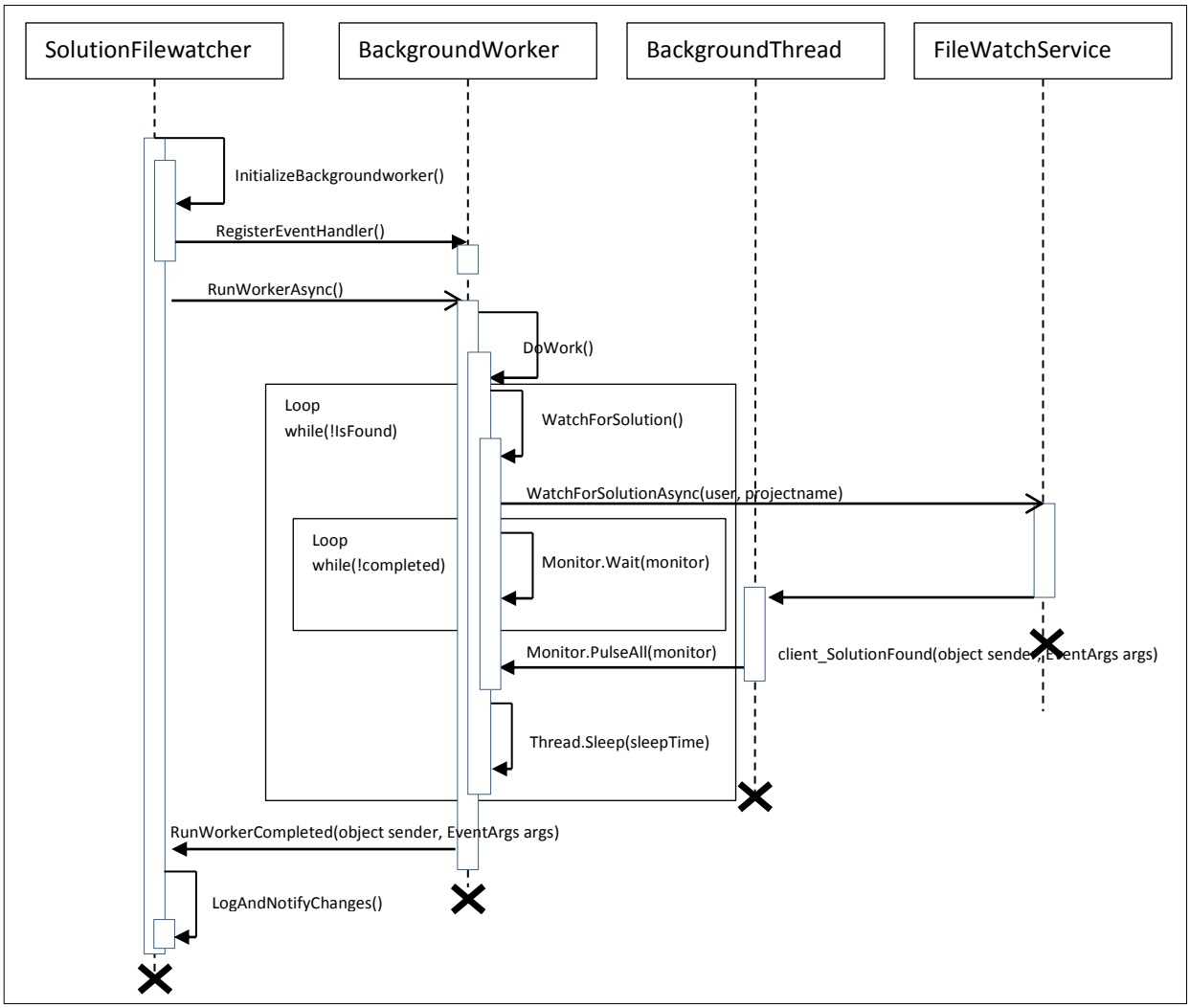

Abbildung 59: SolutionFileWatcher Thread-Modell

Der SolutionFileWatcher hat die Aufgabe, in einem Zeitintervall von vier Sekunden zu überprüfen, ob die Berechnung eines bestimmten Projekts auf dem Cloud-Backend abgeschlossen und somit eine Lösungsdatei vorhanden ist. Um das UI nicht in seiner Funktion zu beeinträchtigen, wird der Vorgang des Abwartens des Zeitintervalls in einen BackgroundThread ausgelagert.

Für den Ablauf wird das Thread-Elemente Monitor verwendet. Der Monitor es erlaubt, eine Methode so lange zu pausieren, bis ein zweiter Thread eine Nachricht sendet, um das Pausieren zu stoppen.

Der Ablauf ist im Thread-Modell der Abbildung 59 dargestellt und setzt sich wie folgt zusammen:

1. Der UI Thread ruft die asynchrone Methode RunWorkerAsync() auf, welche den BackgroundWorker startet.
2. Der BackgroundWorker wiederum ruft seine Methode WatchForSolution() auf.
3. Hier folgt der asynchrone Aufruf der eigentlichen Service Methode: WatchForSolutionAsync(user, projectname).
4. Nun wird ein Lock auf den Monitor gesetzt so lange gewartet bis dieser per Monitor.PullseAll() aufgeweckt wird und andererseits der boolen „completed“ anzeigt, dass

der Service-Aufruf abgeschlossen ist. Monitor.PulseAll() wird dabei in dem Callback des SolutionFileWatchServices aufgerufen.

5. Treffen beide Bedingungen zu, dann kann die Methode verlassen und zur DoWork()-Methode zurückgekehrt werden.
6. Sofern die Lösungsdatei nicht gefunden wurde, also „IsFound“ nicht „true“ ist, wird mit Thread.Sleep(sleepTime) vier Sekunden lang gewartet (sleepTime = 4000). Anschliessend wir wieder zum Punkt 2 zurückgekehrt. Diese Wiederholung von den Punkten 1-6 erfolgt so lange, bis eine Lösungsdatei gefunden wurde. Sobald dies der Fall ist, wird der BackgroundWoker beendet und der SolutionFileWatcher, also der UI Thread, erhält mit einem CompletedEvent Bescheid, dass der Vorgang abgeschlossen ist und die Lösungsdatei gefunden wurde.

3.4.1.1 Busy Indicator Backgroundworker

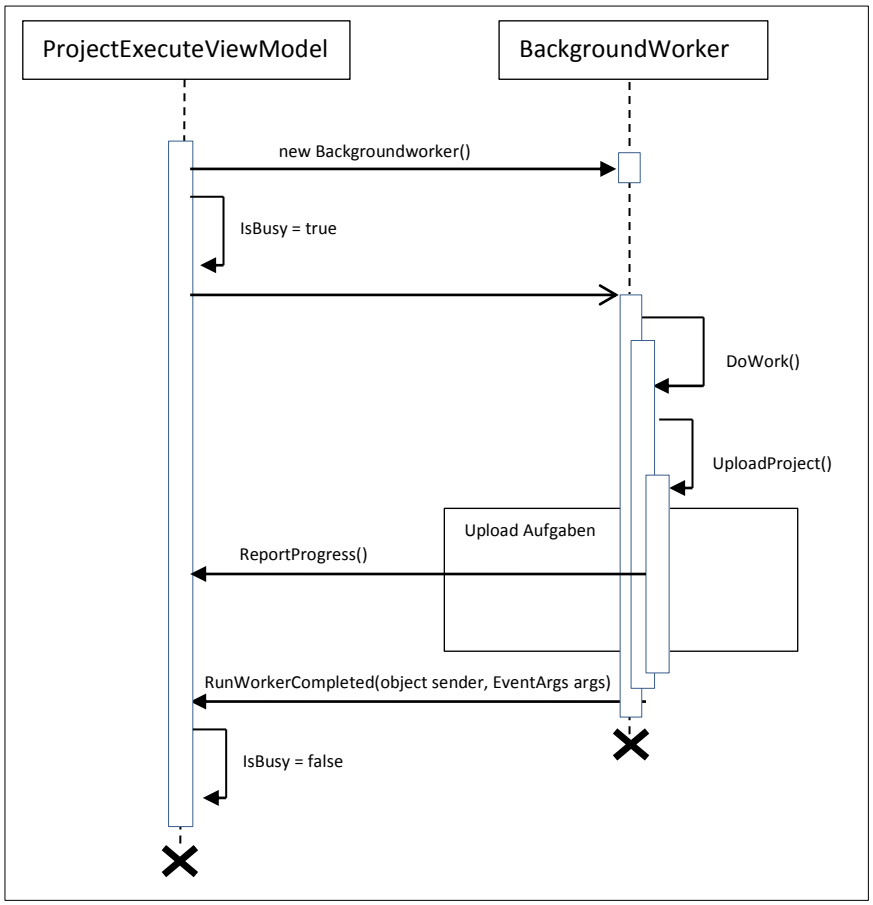

Abbildung 60: Busy Indicator Thread-Modell

Der Busy Indicator erlaubt, wie bereits im Abschnitt 3.3.4.2 des UI Designs beschrieben, bei länger dauernden Aufgaben ein UI-Element anzuzeigen, welches anzeigt, dass die Applikation beschäftigt ist. Die gleichzeitige Aufgabe den Busy Indicator anzuzeigen sowie die Hintergrundaufgabe durchzuführen, bedarf der Trennung dieser beiden Aufgaben in separate Threads. Der Busy Indicator bleibt dabei, als UI Element, im UI-Thread bestehen. Er startet, sobald der Boolean „IsBusy“ auf „true“ gesetzt wird. Für den zweiten Thread wird ein Backgroundworker erstellt, welcher nach dem asynchronen Aufruf die Aufgabe, im abgebildeten Thread-Modell handelt es sich um den Fileupload, ausführt. Die genauen Aufgaben des Fileuploads wurden in dem Modell absichtlich nicht aufgeführt. Wichtig ist, dass laufend Fortschrittsmeldungen an den UI-Thread gesendet werden können und

diese bei Bedarf über den Busy Indicator angezeigt werden können. Dies ist jedoch im gezeigten Beispiel nicht der Fall. Sobald die Upload Aufgaben abgeschlossen sind, erhält der UI-Thread den „RunWorkerCompletedEvent“. Um den somit nicht mehr benötigten Busy Indicator zu beenden, wird der Boolean „IsBusy“ wieder auf „false“ gesetzt.

4 Implementation

In diesem Kapitel wird beschrieben, wie aus einzelnen Entwicklungsprototypen, welche während des Projekts entwickelt wurden, der finale Prototyp resultiert ist. In einem weiteren Abschnitt werden die verwendeten Implementationskonzepte beschrieben.

4.1 Prototyp

Der finale Prototyp besteht aus einer Zusammenstellung der Elemente aus den Entwicklungsprototypen. Die Abbildung 61 stellt dar, welche Elemente im Zusammenhang mit den Entwicklungsprototypen erstellt worden sind. Die Elemente in Grün sind Teil des finalen Prototyps, während die Elemente in Rot im finalen Prototyp nicht mehr benötigt wurden. Dank der Cloud-Schnittstelle, die von Anfang an so geplant wurde, dass sie vom Backend unabhängig ist, benötigte die Umstellung auf Cloudbroker keine zusätzliche Logik für den Silverlight Client.

Zum Prototyp gehört selbstverständlich auch das Front-End, also das UI. Dieses wird jedoch in einem eigenen Kapitel, nämlich im Kapitel 3.3: GUI Design beschrieben.

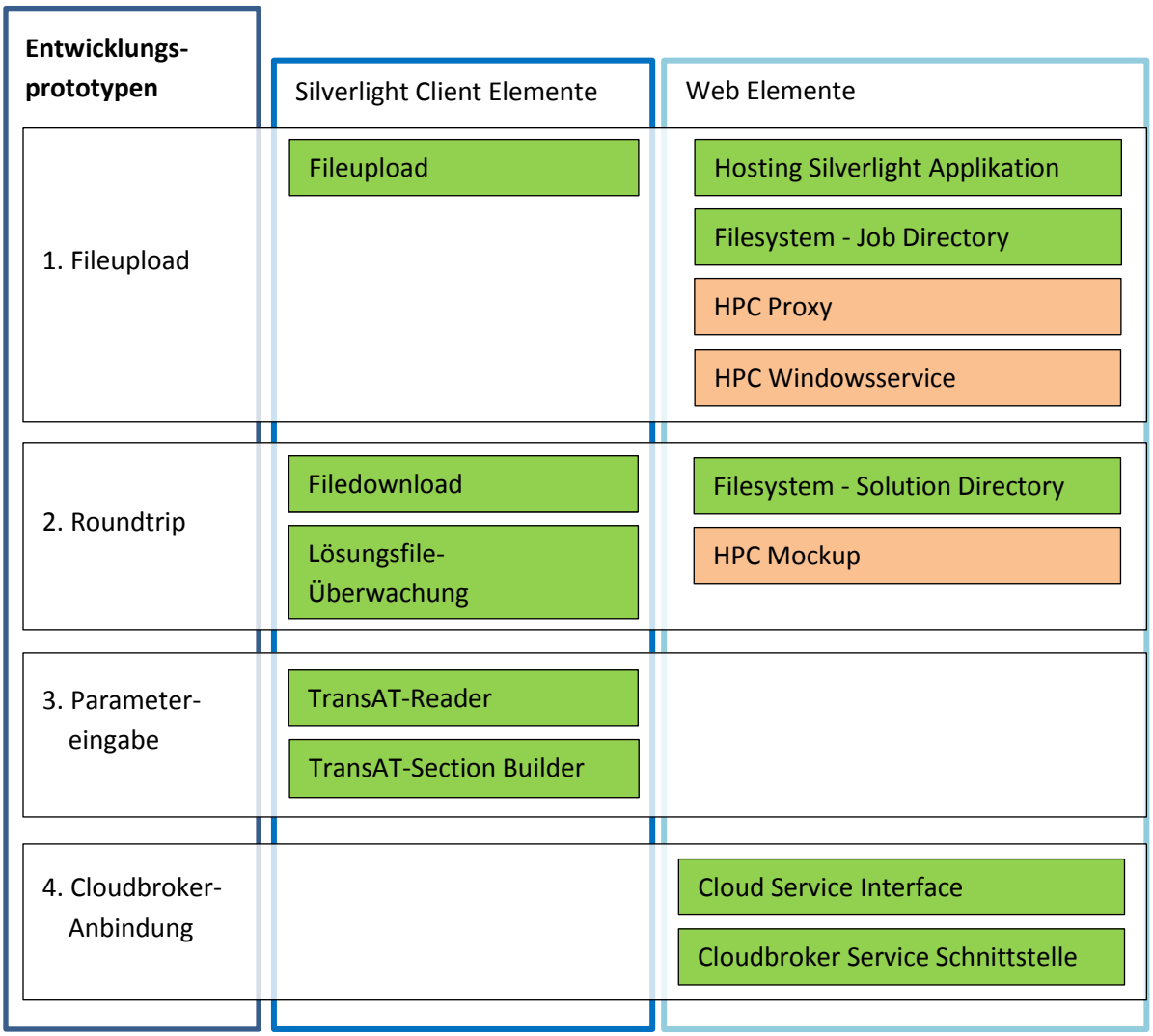

Abbildung 61: Übersicht Prototyp Elemente

4.2 Entwicklungsprototypen

Dieser Abschnitt zeigt die einzelnen, während des Projektes entwickelten, Entwicklungsprototypen auf. Die Entwicklungsprototypen wurden jeweils im Zusammenhang mit der Logik des angestrebten finalen Prototyps geplant. Für die Entwicklungsprototypen wurden Ziele festgelegt, auf welche hin gearbeitet wurde. Die einzelnen Elemente der Entwicklungsprototypen wurden teilweise, aufgrund der ändernden Ziele während dem Entwicklungsprozess, nicht für den finalen Prototypen verwendet.

Die Entwicklung des UIs fand unabhängig vom Entwicklungsprototypen statt.

4.2.1 Entwicklungsprototyp 1: Fileupload

Das Ziel dieses Prototyps war es, einen Architekturdurchstich durch alle Ebenen einer Web- bzw. Cloud-basierten Architektur zu erlangen. Folgende Punkte wurden dabei als Teilziele gesetzt:

- Verbindung zum HSR HPC herstellen
- Upload von Input-Daten eines Simulationsprojekts

4.2.1.1 Resultat

Der Fileupload vom Silverlight Client auf den Webserver erfolgte problemlos. Es wurde dazu ein WCF Webservice angewendet. Da es sich bei den heraufzuladenden Dateien um sehr grosse Datenmengen handeln könnte, wurde der Upload so implementiert, dass die Dateien aufgeteilt und somit in Blöcken heraufgeladen werden.

Da die Cloudbroker-Infrastruktur zum Zeitpunkt des Erstellens des Prototyps noch nicht bereit war und die Priorität auf den HPC gesetzt wurde, wurde dieser Prototyp mit dem HPC-Backend realisiert.

Es wurde festgestellt, dass der Zugriff auf den HPC der HSR relativ kompliziert ist. Dies liegt daran, dass die Infrastruktur des HPCs noch nicht genügend für die Benutzung durch externe Kunden ausgebaut ist. Folgende Probleme liegen vor:

1. Die Authentifizierung erfolgt über das HSR-Active-Directory, d.h. nur für Benutzerkonten der HSR und nicht für externe Kunden.
2. Der HPC liegt im HSR LAN, also hinter der Demilitarized Zone[5] (DMZ) des HSR-Netzwerks und ist so nicht von aussen erreichbar. Dies ist aus sicherheitstechnischen Gründen so.
3. Auf den HPC kann auch nicht vom Webserver aus zugegriffen werden, obwohl dieser innerhalb der DMZ liegt.
4. Jeder Benutzer der auf den HPC zugreifen kann hat so viele Rechte, dass er z.B. „format C:\“ ausführen könnte. Das heisst, er kann das gesamte System löschen.
5. Der TransAT-Simulations-Kernel ist (trotz früherer Ankündigung) noch nicht auf Windows portiert worden.

4.2.1.2 Entscheid

Da zum Zeitpunkt des Entscheids (Projektwoche 3) noch keine Alternativen zum HPC zur Verfügung standen, wurde beschlossen, die nötigen Schritte zu veranlassen um den HPC von aussen her ansprechen zu können. Diese Schritte werden in den folgenden Abschnitten beschrieben.

Um den ersten Prototyp so einfach wie möglich zu halten, wurde auf die Interaktion mit dem vorerst geplanten Webserver von Ascomp verzichtet, da dieser keinen wesentlichen Vorteil gebracht hätte.

[5] http://de.wikipedia.org/wiki/Demilitarized_Zone

Im Gegenteil: Der Mehraufwand hätte einen weiteren externen Server und somit auch eine weitere potenzielle Fehlerquelle in der Architektur hervorgerufen. Die Funktion des Ascomp-Webservers, der das Hosting der Silverlight-Applikation und das Zwischenspeichern der Projektdaten übernommen hätte, wurde somit auf den Proxy im HSR-Netz verlagert. Die genaue Funktion dieses Proxys wird ebenfalls in den folgenden Abschnitten erklärt.

4.2.1.3 Problematik

Die wohl grösste Herausforderung war die Problematik mit dem Netzwerk bzw. mit den Firewalls und den Sicherheitsrichtlinien der HSR. In der Abbildung 62 ist zu sehen, dass einzig die Kommunikation und der Datenaustausch von der DMZ zum HSR-LAN ein Problem darstellt. Es ist aus Sicherheitsgründen nicht möglich von der DMZ aus in das HSR LAN zu gelangen. Umgekehrt ist es jedoch möglich, vom HSR LAN in die DMZ zu gelangen.

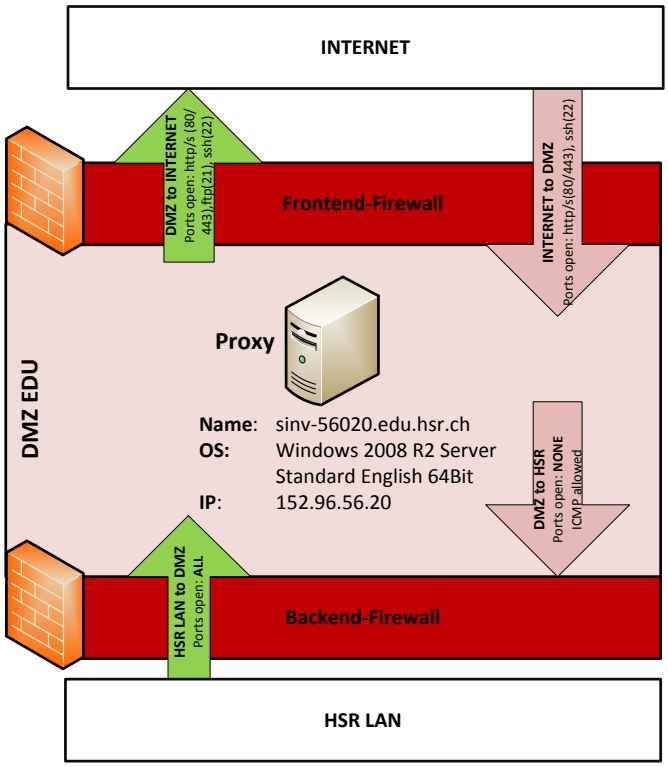

Abbildung 62: HSR Firewalls

4.2.1.4 DMZ-Backend-Firewall umgehen

Der erste Schritt war die Firewall der DMZ zu umgehen. Dies wurde erreicht, indem ein virtueller Windows Server 2008 R2 in die DMZ gestellt wurde, welcher ab nun den „Proxy“ des HPCs in der DMZ darstellt. Des Weiteren musste noch eine Firewall Regel hinzugefügt werden, welche es erlaubt, vom HSR LAN auf eine Windows-Freigabe in der DMZ zuzugreifen. Diese Öffnung der Firewall stiess bei den Informatikdiensten der HSR zuerst auf wenig Zustimmung. Nach einiger Überzeugungsarbeit wurde jedoch erreicht, dass eine Regel erstellt wurde. Diese wird jedoch am 03.03.2012, also nach Beendigung der Bachelorarbeit, wieder gelöscht. Somit handelt es sich hiermit um keine dauerhafte Lösung.

4.2.1.5 HPC-Proxy (Windows Server 2008 R2 - SINV-56020)

Der HPC-Proxy ist nicht einfach nur ein Stellvertreter des HPCs. Er beherbergt noch diverse andere Services, wie z.B. einen ISS[6] oder einen Fileshare. Diese Dienste werden benötig, um seine Funktion als Tauschplattform ausführen zu können.

Der IIS wird benötigt, um die erforderlichen Webservices sowie die Silverlight-Applikation zu hosten.

Der Fileshare-Service des IIS dient zur Kommunikation mit dem realen HPC-Server. Dabei bleibt noch zu erwähnen, dass für den Austausch von Dateien ein zusätzlicher lokaler Benutzer (Ascomp) auf dem Proxy erstellt wurde, welcher neben dem Administrator der einzige ist, welcher Zugriff auf die Ordner Freigabe hat.

4.2.1.6 Projektdatenaustausch Silverlight-Client mit HPC-Proxy

In einem nächsten Schritt sollte es möglich gemacht werden, dass der Silverlight-Client die Projektdaten auf den Proxy laden kann.

Die Sicherheitsrichtlinien der HSR erlauben einem nur Verbindungen vom Internet zur DMZ über die Ports http/s(80/443) und ssh(22). Somit wurde auf dem Client ein WCF Webservice (FileService) geschrieben, welcher die Projektdaten über das http-Protokoll auf den Proxy lädt und dazu gleich schaut, dass die Projektdaten auch im richtigen Directory abgelegt werden. In diesem Fall wäre dies das „Job Directory“.

4.2.1.7 Fileshare Service – „Job Directory“

Das „Job Directory“ ist eine Windows-Freigabe auf welche nur der lokale Benutzer Ascomp Zugriff hat. Dies wurde so entschieden, um eine minimale Sicherheit zu gewährleisten. Der genaue UNC[7]-Pfad auf das „Job Directory“ ist „\\152.96.56.20\TransAtUI\Job“. Dieser UNC-Pfad ist essenziell für die Kommunikation mit dem HPC-Server hinter der DMZ.

4.2.1.8 Windowsservice – Holt neuen Job

Auf dem HPC-Server läuft der Windowsservice „IIS Filewatcher“. Dieser ist dafür verantwortlich, immer die neusten Projektdaten, also die „Jobs“ für den HPC vom HPC-Proxy zu holen und dem HPC zu übergeben.

Bei der Entwicklung des Windowsservice mussten die folgenden Punkte beachtet werden.

1. Der Windowsservice selbst läuft unter einem lokalen Benutzer auf dem HPC. Welcher Benutzer das ist, entscheidet der Administrator des HPC und ist somit vom Bachelor-Team nur beschränkt beeinflussbar.
2. Der Windowsservice kann nur die Pfade bzw. Windowsfreigaben überwachen, welche UNC konform sind.
3. Der HPC-Proxy ist nicht mit dem Active-Directory der HSR verbunden sondern kennt nur die Benutzer, die lokal auf dem HPC-Proxy registiert sind.

Um den ersten Punkt zu berücksichtigen, musste man nur den HPC-Administrator über das Verhalten des „IIS-Filewatcher“ aufklären.

[6] http://www.iis.net/overview
[7] http://de.wikipedia.org/wiki/Uniform_Naming_Convention

Der zweite Punkt wurde schon durch die Umgehung der Firewall bzw. durch den Fileshare-Service gelöst. (siehe Kapitel DMZ-Backend-Firewall umgehen) Um den Serverbetrieb möglichst wenig zu beeinflussen wurde für die Verbindung zur der Windows-Freigabe der HPC-Proxy der Befehl „net use“ verwendet, was zur Folge hat, dass die verbundene Windows-Freigabe im Explorer des HPC-Server nicht ersichtlich ist.

Der dritte Punkt war etwas komplizierter zu realisieren, denn man musste sich vom HPC-Sever aus mit den Benutzerinformationen des am HPC-Proxy lokal registrierten Benutzer Ascomp authentifizieren.

Da das NET-Framework eine solche Authentifizierung nicht von Haus aus unterstützt, musste auf die „NetApi32.dll“ zurückgegriffen werden. Die „NetApi32.dll“ wurde mit Hilfe der „P/Inovoke“ Technik für das .Net-Framework zugänglich.

Jetzt konnte man sich vom HPC-Server aus auf den HPC-Proxy verbinden und sich dort als lokalen Benutzer Ascomp authentifizieren.

4.2.1.9 Windowsservice – „Konfigurationsmöglichkeiten“

Es gibt zwei Möglichkeiten den Service zu konfigurieren, entweder per XML-Datei oder per Startparameter. Grundsätzlich ist zu empfehlen, den Service per XML-Datei zu konfigurieren. Die Startparameter sind eher für schnelle Änderungen oder Testzwecke gedacht. Eine Übersicht über die Konfigurationsmöglichkeiten ist im Anhang des Berichts zu finden.

4.2.2 Entwicklungsprototyp 2: Roundtrip

Das Ziel Roundtrips war die Erweiterung Fileuploads aus Prototyp 1. Dabei soll der Vorgang simuliert werden, wie eine Job-Datei dem Solver gesendet wird, dieser die Datei berechnet und das Resultat, also die Solution-Datei, wieder an den Client zurücksendet.

4.2.2.1 Resultat

Zum Zeitpunkt des Erstellens vom Roundtrip (Woche 6) war die Portierung des Solvers auf den HPC der HSR noch nicht im Gange. Weiter stand auch noch keine Schnittstelle zum Cloudbroker-Interface zur Verfügung, was eine Alternative zum HPC-Cluster gewesen wäre. Es sollte auch der hohe Aufwand, der bis anhin betrieben wurde, um einen Zugang zum HPC-Cluster zu garantieren, nicht umsonst gewesen sein. Aus diesem Grund wurde beschlossen einen Mock-up für den HPC zu schreiben.

4.2.2.2 Windowsservice-Roundtrip

In der Abbildung 63 ist die Erweiterung ersichtlich, welche bis auf die Berechnung auch funktioniert.

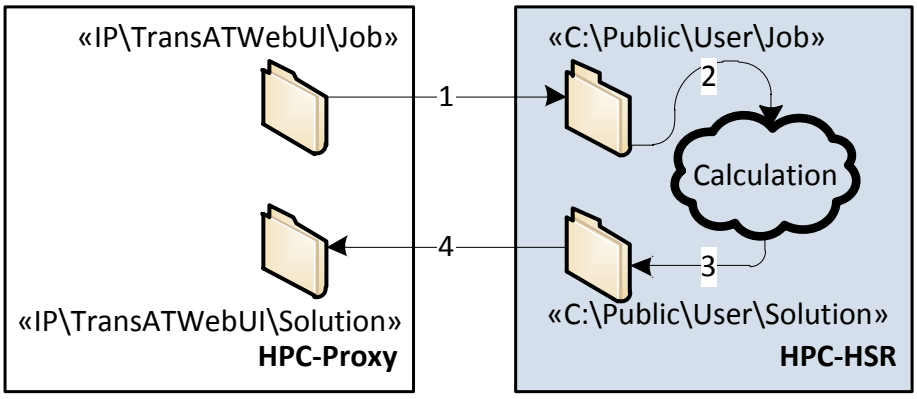

Abbildung 63: Roundtrip-Erweiterung

1. Projektdaten werden vom Windowsservice „IIS Filewatcher“ von HPC-Proxy „Job“-Ordner in den „Job“-HPC verschoben. Die Namenskonvention für die Job-Dateien ist dabei die Zusammenstellung aus Benutzernamen, Projektnamen und dem Auftrag (Job) (benutzer_projektname_job.zip).
2. Sobald alle Projektdaten fertig geladen sind, fängt der HPC mit seinen Berechnungen an.
3. Nach der Berechnung werden die Lösungs-Dateien in ein Zip (benutzer_projektname_solution.zip) gepackt und in den „Solution“-Order des HPC verschoben.
4. Das Zip wird nun in den „Solution“-Ordner des HPC-Proxy verschoben. Falls schon solch eine Datei vorhanden ist, wird diese gelöscht und durch die neue ersetzt. Die zurückgebliebenen Projektdaten werden ebenfalls gelöscht.

4.2.2.3 Mock-up-Konzept

Der ganze HPC-Mock-up wird auf dem Bachelor-PC (PIN1206010) gehostet. Das heisst, es wird dort der Windowsservice „IIS-Filewatcher“ installiert. Auch für diesen Computer gibt es eine spezielle Firewall-Regel, wie zuvor für den HPC, damit er auf die Windows-Freigaben des HPC-Proxy zugreifen kann. Somit ist die gleiche Netzwerkstruktur gewährleistet.

Wie schon zuvor kurz erwähnt, gab es zum Zeitpunkt des Erstellens des Mock-up noch keinen TransAT-Solver für Windows, also für den HPC. Aus diesem Grund wurde für den Roundtrip ein TransAT-Beispielprojekt verwendet, welches bereits gelöst wurde und somit zugehörige Resultatdateien hatte. Dieses Projekt wurde für den Mock-Up als Lösungsprojekt verwendet, d.h. egal welches Job-Projekt der HPC-Mock-up bekommt, er sendet das bereits gelöste Beispielprojekt zurück.

Um die Berechnung eines Projektes zu simulieren, nimmt der Mock-Up das eingegangene Job-Projekt entgegen, dann wartet er eine bestimmte Zeit und kopiert letztendlich die bestehende Lösungsdatei in den Solution-Ordner.

Der Mock-up selber ist ein kleines Konsolenprogramm (Abbildung 64) welches seine wichtigsten Aktivitäten auch gleich über die Konsole ausgibt.

```
Strat mocking HPC
=================
Sourcepath:      C:\Users\Public\Job
Destinationpaht: C:\Users\Public\working
Workingpath:     C:\Users\Public\Solution
Wait for:        5 sek.

Finish with 'q'...

Waiting finish loading..
Waiting for 5 sek
custumer_project_solution.zip
Deleting existing C:\Users\Public\Solution\custumer_project_solution.zip
Moving from C:\Users\Public\working\solution.zip to C:\Users\Public\Solution\custumer_project_solution.zip
Deleting C:\Users\Public\working\custumer_project_job.zip
Finish moving
Ready for the next job...
```

Abbildung 64: Mock-up Ausgabe

Beim Start des Mock-ups wird davon ausgegangen, dass die angegebenen Ordner und Dateien existieren. Falls nicht, wird eine Fehlermeldung ausgegeben und der Mock-up wird beendet.

4.2.2.4 Mock-up Roundtrip

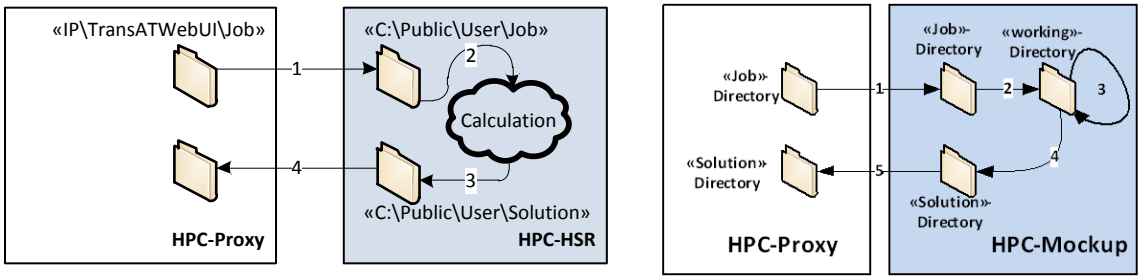

Abbildung 65: HPC-Mock-up

Das Ziel war, einzig den HPC durch einen anderen Server zu ersetzen. Somit ist in der Abbildung 65 ersichtlich, dass sich am eigentlichen Roundtrip nichts geändert hat.

4.2.2.5 Mock-up Konfiguration

Grundsätzlich müssen vor dem Start des Mock-ups keine zusätzlichen Einstellungen getätigt werden, damit er läuft. Man kann aber, sofern gewünscht, alle wichtigen Parameter konfigurieren. Es gibt zwei Möglichkeiten dies zu tun:

1. Falls man die Standardwerte nicht überschreiben möchte, so lassen sich alle Werte per Startargument für die Laufzeit überschreiben. Die komplette Liste aller Werte findet man in der Tabelle Mock-up Parameter.
2. Falls man jedoch die Werte für immer ändern will, kann man dies tun, indem man mit einem beliebigen Texteditor die Datei „HPCMockup.exe.config“ öffnet und die Werte nach eigenem Bedürfnis anpasst. Die Datei befindet sich im gleichen Verzeichnis wie die Datei „HPCMockup.exe“. Die Datei „HPCMockup.exe.config“ sollte selbstbeschreibend sein. Hier ein kleines Beispiel des „Source Path“ mit seinem Standartwert „C:\Users\Public\Job“.

```
<setting name="JobPath" serializeAs="String">

      <value>C:\Users\Public\Job</value>

</setting>
```

Konsolen-argument	XML	Standartwert	Name	Beschreibung
-t	Time	5	Time	Die Zeit die gewartet werden soll
-jp	JobPath	C:\Users\Public\Job	Job-Path	Hierhin werden die Job-Dateien kopiert. (benutzer_projektname_job.zip)
-sp	SolutionPath	C:\Users\Public\Solution	Solution-Path	Hierhin werden die Solution-Dateien kopiert. (benutzer_projektname_solution.zip)
-wp	WorkingPath	C:\Users\Public\working	Working-Path	Hier wird das bestehende „SolutionFile“ gespeichert und die Projektdaten zwischen gelagert.
-sf	SolutionFile	solution.zip	Solution-File	Diese Datei wird immer als Projektlösung geschickt. (benutzer_projektname_solution.zip)

Tabelle 4.1: Mock-up Parameter

4.2.3 Entwicklungsprototyp 3: Parametereingabe

Das Ziel dieses Prototyps war es, die Parameter, die in dem WebUI eingegeben wurden, in die entsprechende Projektdatei zu schreiben bzw. auszulesen.

4.2.3.1 Resultat

Für das Lesen der Parameter in Projektdateien wurde ein Reader geschrieben. Dieser dient als Hilfsklasse für die ViewModels, welche dem UI die Parameterwerte zur Verfügung stellen und Eingaben des Benutzers entgegennehmen.

4.2.3.2 Reader

Der Reader dient hauptsächlich dazu, um zu zeigen, dass es möglich ist die Datei „transat.inp“ zu lesen und die Metadaten über ein Projekt auszulesen.

Der Reader selbst ist so implementiert, dass er die Sektionen anhand der Steuerzeichen

- & - Start einer Sektion
- \r\n/ - Ende einer Sektion

erkennt. Die Key-Value-Paare die sich zwischen den Sektionen befinden gibt er dann als Dictionary wieder zurück.

Die Kommentare

- ! hier ist ein Kommentar bis zum Ende der Zeile

löscht er während des Lesen raus.

Für genauere Beschreibungen der Datei „transat.inp“ und deren Sektionen wird an dieser Stelle auf das offizielle Solver-Manual, Kapitel 4.2 der Firma Ascomp verwiesen.

4.2.3.3 Sektions Builder

Um wieder in die Datei „transat.inp“ zu schreiben, wurde die abstrakte Klasse TransATSectionBuilder geschrieben, welche den Ablauf der Bildung einer Sektion beinhaltet.

Für jede Sektion muss dann einfach eine Klasse implementiert werden welche die folgenden drei Methoden enthält:

- InitDefaultValues
- SetName
- IsValidCombination, hier wird überprüft ob die Eingaben korrekt sind

Zu diesen Methoden müssen dann noch zusätzlich die Key von den Key-Value-Paaren als public const definiert werden, damit man später auf diese zugreifen kann und somit das Schreiben sowie auch das Lesen vereinfachen wird.

4.2.4 Entwicklungsprototyp 4: Cloudbroker-Anbindung

Ab dem 02.11.2011 (Projektewoche 7) war die Cloudbroker-Infrastruktur bereit. Trotzt der Aufwendung von viel Zeit und Energie für das HPC-Backend blieb der Durchbruch leider aus. Aus diesem Grund wurde nun auf die Cloudbroker-Infrastruktur umgeschwenkt.

4.2.4.1 Resultat

Das Ergebnis ist die voll funktionstaugliche Anbindung des WebUI Front-Ends an die Cloudbroker-Infrastruktur.

4.2.4.2 ICloudService

Das ICloudService-Interface definiert die minimale Anforderung, welche die Silverlight-Applikation „TransAt WebUI“ benötigt. Das Interface muss von den Webservices implementiert werden, welche sich auf ein Backend verbinden. In diesem Fall wäre dies das Cloudbroker-Backend und der dazu gehörige CloudbrokerService.

In der Abbildung 66 sind die Funktionen des ICloudService Interfaces ersichtlich. Diese Funktionen werden von den Webservices der Silverlight-Applikation aufgerufen. Da alle Webservices, die sich auf ein Backend verbinden, dieses Interface und somit diese Funktionen implementieren, ist die einheitliche Schnittstelle gewährleistet. Das heisst, für die Silverlight-Applikation ist es nicht von Bedeutung, welches Backend hinter dem Service steckt, da die Aufrufe immer gleich bleiben.

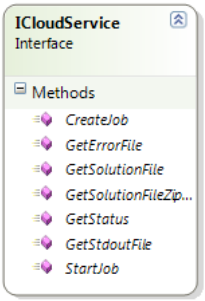

Abbildung 66: Funktionen des ICloudService

4.2.4.3 CloudbrokerClient

Der CloudbrokerClient ist eine Erweiterung des normalen Webclients, welcher vom .Net-Framework zu Verfügung gestellt wird.

Die Erweiterung umfasst folgende Punkte:

- Die komplette Einbindung der RESTfull API von Cloudbroker (siehe Anhang Cloudbroker API)
- Die Authentifizierung bei Cloudbroker

- SSL-Zertifikat Exceptionhandling, da Cloudbroker keine signierten SSL-Zertifikate besitzt.
- Die De-\ Serialisierung der Objekte zu XML-String und wieder zurück

4.2.4.4 Cloudbroker Service

Der Cloudbroker Service implementiert nicht nur die vom ICloudService diktierten Funktionen, sondern ergänzt den Cloudbroker-Client noch um die folgenden Funktionalitäten:

- Das Herunterladen der einzelnen Daten von den konkreten Clouds wie z.B. Amazon
- SSL-Exceptionhandling für die Clouds, sprich dass der Webclient alle akzeptiert
- Das archivieren der einzelnen Lösungsdateien zu einer einzelnen Lösungs-Archivdatei (Zip)

4.2.4.5 Übersicht

In der **Error! Reference source not found.** ist eine Übersicht über die in den vorherigen Abschnitten erklärten Komponenten gegeben.

Abbildung 67: Übersicht des Zusammenspiels der einzelnen Komponenten

4.3 Konzepte

4.4 Implementationskonzepte

4.4.1 Silverlight Client

Der Silverlight Client orientiert sich am MVVM-Pattern von Microsoft. Dabei wurde das Pattern mit einem zusätzlichen Service-Abschnitt ergänzt, da dieser im Falle dieser Applikation einen sehr bedeutenden Teil ausmacht. Da es sich bei den verwendeten Services nicht um Datenservices, sondern um logische Services handelt, ist der Service Layer nicht wie üblich unterhalb des Models angelegt, sondern ist direkt mit dem View Model verbunden.

In der Abbildung 68 ist die Trennung zwischen Front-End in der View, Front-End Logik im ViewModel und weiteren, vom Front-End unabhängigen Funktionen im Service Layer und im Data Model ersichtlich. Dank dieser strikten Trennung kann der Client sehr einfach erweitert werden. Wird zum Beispiel eine View angepasst, so muss unter Umständen noch das zugehörigen View Model angepasst werden, das Data Model und der Service Layer sind jedoch von der Änderung nicht betroffen.

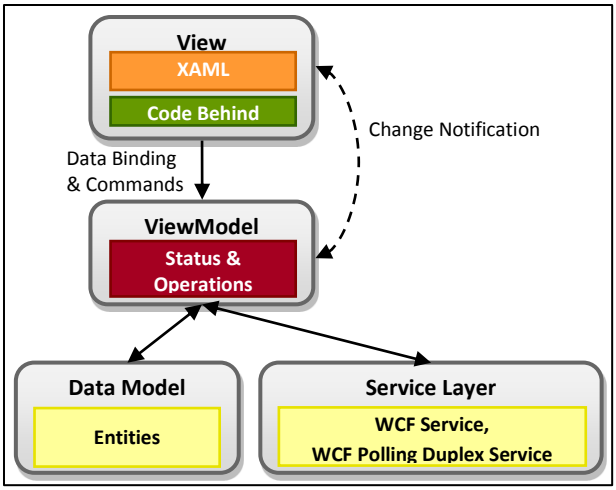

Abbildung 68: Aufteilung des Clients nach MVVM

5 Testing

5.1 Allgemein zum Test

Das Ziel war es nicht alle nur möglichen Eventualität und Funktionen des Projektes zu testen für dies reichte schlicht die Zeit nicht. Somit hat sich das Testen auf das Cloud-Backend und auf den Parser fokussiert. Des Weitern ist das Testen von Silverlight-Applikationen noch nicht so ausgereift.

Es gab insgesamt drei Test-Projekte TestUtilits, CloudbrokerAPITest und CloudbrokerServiceTest, die einzelnen Projekte werden nachfolgend besprochen.

Bemerkung
Wegen mangelnder Erfahrung im Umfeld von .Net und Visual Studio 2010 des Testschreibers, sind die Tests sehr Maschinenabhängig, entgegen seiner Überzeugung.

5.1.1 TestUtilits

In diesem Projekt wurden der Reader, Builder, die Extensionmethoden und das WebUI.Utilities getestet.

Hier zu erwähnen ist um die das Projekt WebUI.Utilites zu testen wurde es als normales Libary-Projekt entwickelt und getestet. Da es offiziell von Microsoft noch eine Unit-Test für Silverlight gibt.

5.1.1.1 CloudbrokerAPITest

In diesem Test-Projekt wurde der Verbindungsaufbau vom CloudbrokerClient zum Cloudbroker sowie die richtige Serialisierung bzw. Deserialisierung der Objekte getestet.

5.1.1.2 CloudbrokerServiceTest

In diesem Test-Projekt wurde der CloudBrokerService getestet. Und zwar wurde geschaut das ob er alle Funktion des ICloudService Interface richtig implementiert hat und ob das Zippen auch klappt.

Speziell zu erwähnen ist hier das der Service bei den Test lokal gestartet wird aber ansonsten gleich mit dem Cloudbroker-Backend kommuniziert.

6 Resultat und Weiterentwicklungen

6.1 Resultat

Das Ziel des finalen Prototyps ist die wesentlichen Elemente des Simulationsworkflows mit der web- und cloud-basierten Architektur zu realisieren.

Das Resultat umfasst folgende Aspekte:

- Projektverwaltung: Übersicht, Erzeugung, Editieren, Kopieren und Löschen von Simulationsprojekten pro Benutzer.
- Up- und Download von vollständigen Simulationsprojekten
- Upload von einzelnen Input-Dateien für ein Simulationsprojekt
- Ansicht und Editierung von Input-Parameter für die Simulation
- Starten und Ausführen der Simulation auf einem Cloud-Backend (Cloudbroker)
- Download der gelösten Simulationen
    - Eine oder mehrere Lösungen zusammen aus der Projektverwaltung
    - Direkt aus einem geöffneten Projekt

Dies illustriert die wesentlichen Aspekte der Simulation und die generelle Machbarkeit eines rein web- und cloud-basierten Fluid Dynamics Simulationssystems.

Die Implementation dieses Prototyps hat aber auch Einschränkungen:

- Keine Persistenz: Die Projekte werden noch nicht serverseitig gespeichert. Hier wäre eine Datenbank für die Projekt-Metadaten sowie eine File-Struktur für die Projekt-Dateien nötig.
- Kein Multi-User Login. Hier werde eine serverseitige Benutzerverwaltung in einer Datenbank nötig
- Monitoring der laufenden Simulationen
- Kein Abbruch einer laufenden Berechnung
- Keine Input Visualisierung: Nicht zwingend nötig, weil man die Daten vorgängig in ParaView darstellen kann. Mit SL5 könnte man unter Umständen einen hardwarebeschleunigten Viewer einsetzen
- Keine Output Visualisierung: Hier verwendet man ParaView
- Der HPC der HSR kann aufgrund der Firewall-Einschränkung und der ausstehenden Portierung des TransAT-Kernels noch nicht verwendet werden

Diese Einschränkungen sind hauptsächlich eine Frage des Aufwands und der Quantität bzw. optionale Features, die keinen relevanten Einfluss auf die Machbarkeit haben.

6.1.1 Erfüllte Anforderungen

In der Tabelle sind die aus der Analyse hervorgegangenen Anforderungen aufgelistet. Es lässt sich feststellen, dass

Anforderung	Erfüllungsgrad
Erstellen von Simulationsprojekten	Erfüllt.
Bearbeitung eines Simulationsprojektes mit der Eingabe von bis zu 1000 Parameter	Teilweise erfüllt, Parameter wurden reduziert.
Mapping der UI-Parametereingaben auf das Simulationsprojekt	Teilweise erfüllt, für die reduzierten Parameter ist das Mapping möglich.
Berechnung des Simulationsprojektes	Erfüllt, die Cloudbroker-Schnittstelle wurde erfolgreich angeschlossen.
Anzeige des Berechnungslogs	Nicht erfüllt, das Berechnungslog wird innerhalb des Datei-Archivs mit den Lösungsdateien zurückgegeben.
Anzeige des Resultats mit externem Visualisierungsprogramm	Erfüllt.
Projektverwaltung	Teilweise erfüllt, die Projektverwaltung ist nicht web-basiert und nicht persistent. Die aktuelle Lösung dient einzig als Übergangslösung.
Benutzerverwaltung	Nicht erfüllt.

Tabelle 6.1: Erfüllte bzw. nicht erfüllte Anforderungen

6.1.2 Erfüllte nichtfunktionale Anforderungen

In der Tabelle sind die aus der Analyse hervorgegangenen nichtfunktionalen Anforderungen aufgelistet. Der genaue Inhalt der nichtfunktionalen Anforderungen ist im Kapitel 1: Anforderungs-spezifikation nachzulesen.

Nichtfunktionale Anforderung	Erfüllungsgrad
Zuverlässigkeit	Teilweise erfüllt, die Applikation ist nicht anfällig auf Verbindungsunterbrüche, solange ein Simulations-projekt nicht gerade zum Zeitpunkt des Verbindungs-unterbruchs abgesendet wird. Natürlich würde sich die Situation bei einer allfälligen eingesetzten serverseitigen Datenbank wieder ändern.
Benutzbarkeit	Erfüllt.
Antwortzeiten	In der Testumgebung erfüllt. Über eine andere Umgebung kann keine Aussage gemacht werden.
Ressourcenbedarf	Erfüllt.

Wartbarkeit, Änderbarkeit	Erfüllt. - Beim Silverlight Client wurde darauf geachtet dass, das Zwischenspeichern der Projektdaten so implementiert ist, dass man ohne grosse Änderungen eine Datenbank anbinden könnte. - Das gleiche kann man von der Anbindung zur Cloud-Infrastruktur sagen.
Portierbarkeit	Teilweise erfüllt, unter Linux funktioniert die Applikation trotz neustem Moonlight-Plug-In (Apr 06 2011) nicht. OS-X wurde nicht getestet.
Vertraulichkeit und Datenintegrität	Teilweise erfüllt. Die Kommunikation zwischen der Cloudschnittstelle und dem Cloud-Infrastruktur (Cloudbroker) geht über HTTPS. Die restliche Kommunikation ist unverschlüsselt.

Tabelle 6.2: Erfüllte bzw. nicht erfüllte Nichtfunktional Anforderungen

6.1.3 Erfüllte Use-Cases

In der folgenden Tabelle sind die Use-Cases mit ihrem Erfüllungsgrad aufgelistet.

Use Cases	Erfüllungsgrad
UC01: Simulationsprojekte anlegen und verwalten	Erfüllt.
UC02: Parameter bearbeiten	Erfüllt. Absprache mit Ascomp, dass ein reduzierte Anzahl Parameter ausreicht.
UC03: Job an Solver senden	Erfüllt.
UC04: Resultat entgegennehmen	Erfüllt.
UC05: Login	Nicht Erfüllt.

Tabelle 6.3: Erfüllte bzw. nicht erfüllte Use Cases

6.1.4 Erkenntnisse

Da es sich bei der Bachelorarbeit um eine Machbarkeitsstudie handelt, war es das Ziel, auch Unklarheiten bzw. Offene Fragen zu beantworten.

Im folgenden Kapitel werden die wichtigsten Fragen beantwortet. Des Weiteren wird noch ein kurzer Blick auf die möglichen Weiterentwicklungen geworfen.

6.1.4.1 Silverlight

Frage	Status	Antwort
Ist es möglich Silverlight als Client-Technologie zu verwenden?	beantwortet	Ja
Ist es möglich ein 3D-Rendering zu	beantwortet	In Silverlight 4 nicht machbar

implementieren		
Können grosse (1GB) File hoch/runtergeladen werden?	Nicht beantwortet	Grosse Files können nicht im IsolatedStorage abgelegt werden, was jedoch notwendig ist für die Applikation. Der Up-/Download an und für sich würde funktionieren
Können die TransAT Projektdateien ausgelesen, bearbeitet und wieder eingelesen werden?	beantwortet	Ja

Tabelle 6.4: Silverlight Erkenntnisse

Es war möglich, eine Silverlight-Applikation zu erstellen, welche alle Aspekte einer web- bzw. cloud-basierten Applikation abdeckt.

6.1.4.2 HSR / Cloudbroker-Backend

In der untenstehenden Tabelle werden die allgemeinen Fragen aufgelistet, die man sich in Bezug der Machbarkeit einer Cloud fragen sollte. Es gibt sicher noch unzählige weitere Fragen, hier wurden aber nur jene behandelt, die für diese Machbarkeit Studie von Bedeutung waren.

Legende zur folgenden Tabelle:

- O.K. Die Frage konnte beantwortet werden
- NO Die Frage konnte nicht beantworte werden.
- NB Nicht bekannt, wurde nicht herausgefunden.
- AD Active Directory der HSR

Frage	**Status**		**Antwort**	
	HSR	**Cloudbroker**	**HSR**	**Cloudbroker**
Allgemeine Fragen zu Erreichbarkeit				
Ist es möglich irgendeinen Zugang zur Cloud zu bekommen?	O.K	O.K	Ja (AD)	Cloudbroker Login
Ist es möglich die Cloud ausserhalb ihres lokalen Netzwerkes zu erreichen?	O.K	O.K	Nein	Ja
Gibt es eine technologieunabhängige Schnittstelle zur Cloud (wsdl Webservice / RESTFul Webservice)?	O.K	O.K	Die Funktion war nicht aktiviert.	RESTFull-API
Gibt es eine technologieabhängige Schnittstelle?	O.K	O.K	.NET HPC API	Nein

Gibt es ein Job-Monitoring?	NO	O.K	NB	Ja
Kann man eine laufende Berechnung abbrechen?	NO	O.K	NB	Nein
User Management				
Hat die Cloud eine Zugangskontrolle?	O.K	O.K	AD	Ja
Ist die Cloud von „gefährlichen“ Befehlen von seitens Benutzer geschützt?	O.K	O.K	Nein	Ja
TransAT Solver				
Läuft TransAT auf der Cloud?	OK O.K	O.K	Nein	Ja
Ist TransAT von unbefugten Benutzern geschützt?	O.K	O.K	Nein	Ja
Ist es möglich, eine Berechnung mit TransAt zu starten und Lösungen abzuholen?	O.K	O.K	Nein	Ja

Tabelle 6.5: Backend-Vergleich

HPC-HSR die unbeantworteten Fragen

Die Fragen ob einen die Jobs bzw. Berechnungsaufträge überwacht bzw. abgebrochen werden können, wurde nicht beantworte. Diesen Fragen wäre nachgegangen worden, sobald der TransAT-Kernel portiert gewesen wäre.

HPC-HSR technologieunabhängig Schnittstelle

Es gibt zwar auch für den HPC eine technologieunabhängige Schnittstelle. Dabei handelt es sich um eine RESTFull API[8], diese wurde jedoch von den HPC Betreiber nicht aktiviert.

Fehlende Fragen

Es gibt Fragen, die ganz bewusst weggelassen wurden, weil diese zum Zeitpunkt der Arbeit noch keine relevante Rolle spielten. Folgend eine Auflistung der Fragen, die weggelassen wurden:

- Kostenkontrolle bzw. Abrechnung
- Datenschutz
- Konfigurierbarkeit der Berechnungsaufträge, z.B. wie viele Kerne werden benötigt? Soll der Auftrag erst nach einer bestimmten Verzögerung starten? usw.
- Wo werden die Daten der gelösten Projekte abgespeichert?

6.2 Weiterentwicklung

In diesem Kapitel werden die Weiterentwicklungsmöglichkeiten des Prototyps beschrieben und der Aufwand geschätzt. Dabei wird der Fokus nun auf die Weiterentwicklung des Front-Ends gesetzt, da das Cloud-Backend im Bereich der für das Front-End notwendigen Funktionen abgeschlossen ist. Um

[8] http://msdn.microsoft.com/en-us/library/hh560258(v=vs.85).aspx

eine eventuelle Liste von fehlenden Features (unabhängig von Front-End) für das entsprechen Cloud-Backend zu bekommen, wird an dieser Stelle auf die Tabelle 6.5: Backend-Vergleich verwiesen.

Bei der Aufwandschätzung wird von einem erfahren .Net-Entwickler-Team mit Silverlight-Erfahrungen ausgegangen und dass sich das Team vor Beginn der Weiterentwicklung mit der bestehenden Arbeit auseinander. Weiter wird von einer kompletten Microsoft Umgebung ausgegangen, da diese ein optimales Zusammenspiel aller Komponenten garantiert.

6.2.1 TransAT-WebUI Architektur

Auf der Abbildung 6.1 ist eine mögliche Architektur der finalen WebUI Front-End Applikation abgebildet. Dies ist nur ein Vorschlag und nicht abschliessend. Die Hellgrünen Bereiche sind schon zum grossen Teil fertiggestellt. Die Dunkelgrünen sind so weit fertig, jedoch noch weiter ausbaubar.

Da es so viele Möglichkeiten für die Weiterentwicklung des TransAT-WebUIs gibt, sollen die folgenden Beschreibungen als Empfehlung bzw. Vorschlag für die Ascomp für ihr weiteres Vorgehen dienen.

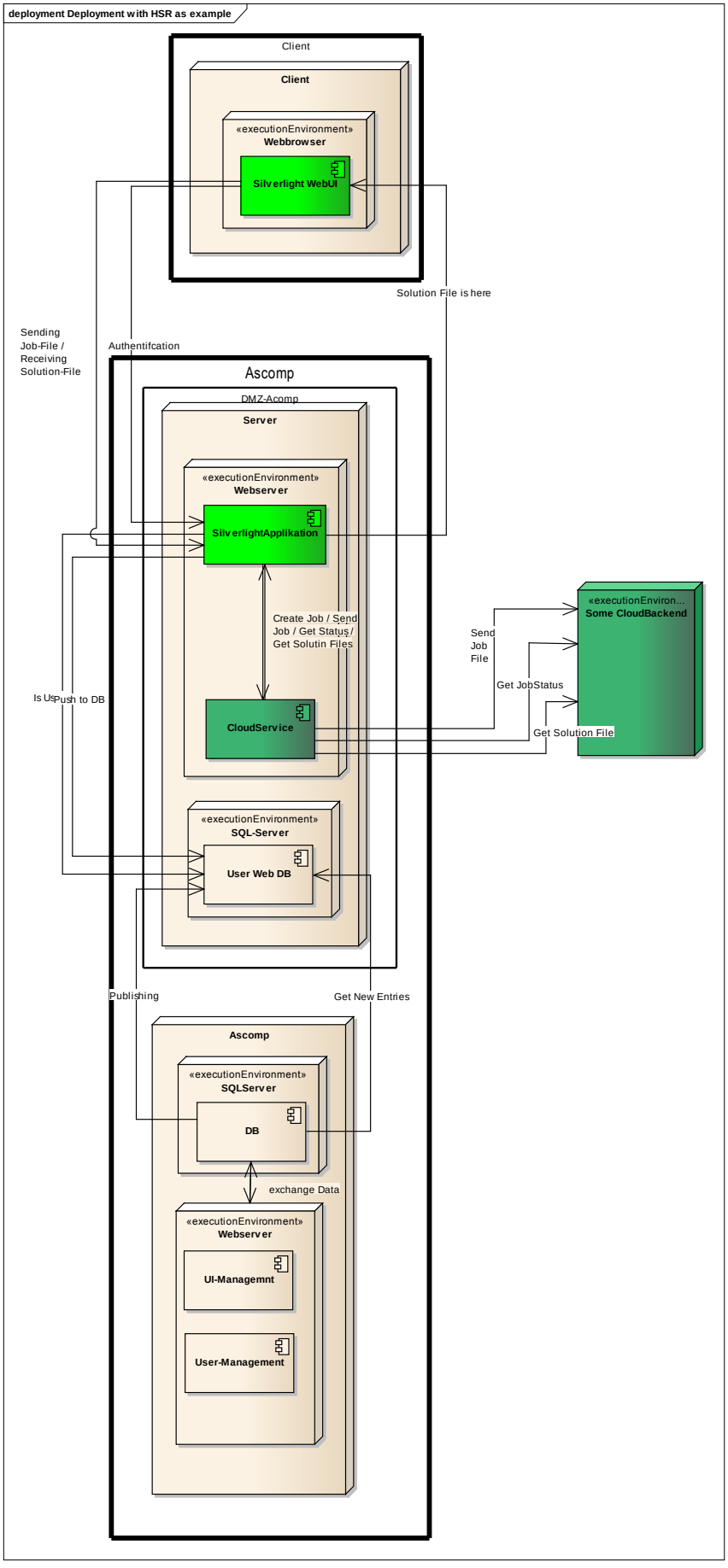

Abbildung 6.1: Mögliche Architektur

6.2.2 Projekt, Benutzer und Lizenz Verwaltung

Als nächster Schritt wird empfohlen, sich Gedanken über die Benutzerverwaltung zu machen. Sprich aus der momentanen Single-User-Applikation eine Multi-User-Applikation zu machen. Die Grundsteine dafür sind bereits gelegt.

Im gleichen Zuge kann man sich auch Gedanken über eine persistente Verwaltung der Simulationsprojekte machen:

- Wo werden die Projekte gespeichert?
- Was wird von den Projekten gespeichert?
    - Nur die Parametereinstellungen oder doch lieber die ganze Datei?
    - Metadaten in DB persistieren und Dateien in einem Filesystem ablegen?

Zur Komplettierung des Ganzen würde es sich auch empfehlen gleich noch die Lizenz-Verwaltung miteinzubinden.

6.2.2.1 Aufwandschätzung

Projekt und Benutzerverwaltung mit der Erstellung einer einfachen Datenbank und einem allfälligen Filesystem und der Einbindung in die bestehende Arbeit (Ansatz für die Anbindung ist bereits gegeben). → **130h**

Lizenz Verwaltung mit Anbindung an die Benutzerverwaltung und allen Sicherheitsrelevanten Aspekten. → **>130h**

6.2.3 UI-Management

Die Ascomp Mitarbeiter erwähnten, dass das TransATUI in Bezug auf die Parametereingabe immer hinter dem TransAT Solver her hinkt, da ständig neue Parameter hinzugefügt werden müssen. Mit dem deklarativen WPF-Framework von Microsoft sollte es möglich sein, eine automatisierte oder wenigstens eine teilweise automatisierte GUI-Generierung anhand der Parameter zu erstellen.

6.2.3.1 Aufwandschätzung

UI-Management so, dass man innerhalb von unter 5 Minuten den neuen Parameter an der richtigen Stelle im GUI eingefügt und getestet hat, im besten Fall würde man es generieren lassen. → **500h (3 Monate)**

6.2.4 Konfigurationen Management

Der Benutzer kann z.B. einstellen wie viele Ressourcen er gerne für einen Berechnungsauftrag haben will und dass der Job erst in drei Stunden starten soll.

6.2.4.1 Aufwandschätzung

Konfigurationsmanagement mit Cloudbroker: Im Kontext zu Cloudbroker sollen alle Konfigurationsmöglichkeiten für einen Berechnungsauftrag, die man beim Cloudbroker-UI zur Verfügung hat, auch im TransAT WebUI zu Verfügung stehen. → **90h**

6.2.5 3D Rendering in der Silverlight Applikation (Silverlight 5)

Wie im Prototyp angedeutet, soll der Benutzer eine 3D Darstellung seiner Objekte, die er unter dem Menupunkt „Mesh“ erstellt, haben.

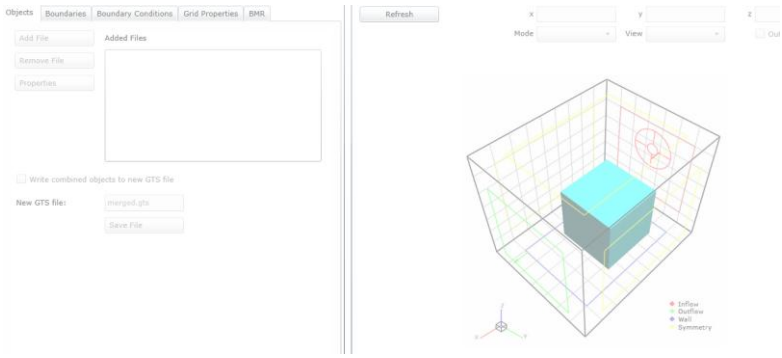

Abbildung 6.2: Mögliche 3D Darstellung im Menupunkt „Mesh“

6.2.5.1 Aufwandschätzung

3D Rendering in der Silverlight Applikation (Silverlight 5), so wie in der Abbildung 6.2 mit Silverlight 5

→ >=100h

Teil III

Projektmanagement

1 Projektplan

1.1 Prozess und Projektaufbau

Der Prozess des Projektmanagements orientierte sich an Scrum[9]. Es wurde jedoch die Freiheit behalten, die Scrum-Strukturen im Verlauf des Projektes anzupassen. Weiter wurde im Verlauf des Projekts auch der Prozess fortlaufend angepasst, da immer wieder Änderungen sowie einige Risiken eingetreten sind.

Die Dokumentstruktur der Projektdokumentation orientiert sich an der Vorlage von Herrn Prof. Stefan F. Keller, Dozent an der HSR.

1.2 Sprint-Einteilung

Orientiert an Scrum wurde die Zeitplanung in Sprints aufgeteilt. Entgegen der Scrum-Vorgabe, dass alle Sprints gleich lang sein müssen, variieren die hier definierten sieben Sprints zwischen einer Zeitdauer von einer bis drei Wochen. Die Aufteilung zusammen mit den Zielen sind in der Tabelle 1.1: Projektplanung aufgeführt. Die Ziele wurden laufend angepasst.

Sprint	Dauer	Von	Bis	Ziele
Sprint 0	1 Woche	19.09.2011	25.09.2011	- Projektplanung - Aufbau der Arbeitsumgebung
Sprint 1	2 Wochen	26.09.2011	09.10.2011	- Einarbeiten in unbekannte Technologien - Grobanalyse - Grobkonzept
Sprint 2	2 Wochen	10.10.2011	23.10.2011	- Entwicklungsprototyp mit Architekturdurchstich - Silverlight Applikation mit Fileupload - Schnittstelle zu HPC
Sprint 3	2 Wochen	24.10.2011	06.11.2011	- Implementierung Roundtrip - Kommunikationskonzept
Sprint 4	2 Wochen	07.11.2011	20.11.2011	- HPC Mockup - Front-End Implementation - Priorisierung der Vertiefungsthemen

Sprint 5	3 Wochen	21.11.2011	11.12.2011	- Implementierung der Vertiefungspunkte:
				  - Einbindung von Cloudbroker
				  - Ausbau des Front-Ends
				- Testing

Sprint 6	1 Woche	12.12.2011	18.12.2011	- Abschluss Cloudbroker - Refactoring und Bugfixing - Dokumentationen
Sprint 7	1 Woche	19.12.2011	23.12.2011	- Abschluss der Dokumentation

Tabelle 1.1: Projektplanung

1.3 Meilensteine

Im Projekt wurden verschiedene Meilensteine definiert, die vor allem nach dem Abschluss eines Sprints gesetzt wurden. Da es im Projekt zu vielen Verzögerungen und Änderungen im Projektablauf kam, konnten nicht alle der zu Beginn festgelegten Meilensteine eingehalten werden.

[9] http://de.wikipedia.org/wiki/Scrum

Meilenstein	Datum	Name	Arbeitsresultat
MS1	21.10.2011	Entwicklungsprototyp mit Architekturdurchstich	Aufgrund der Infrastrukturprobleme, die im Bezug zum HPC aufgetaucht sind, konnte der Prototyp erst mit einer Verspätung von einer Woche fertiggestellt werden.
MS2	28.10.2011	Priorisierung	Aufgrund der Verzögerungen rund um den HPC und dem späten Bereitwerden von Cloudbroker wurde die Priorisierung auf den 23.11.2011 verschoben.
MS3	23.12.2011	Abgabe der Bachelorarbeit	

1.4 Wöchentliche Sitzung

Während der Projektdauer wurden wöchentliche Fortschrittsbesprechungen zwischen dem Betreuer und dem Projektteam abgehalten. Zusätzlich wurde auch unregelmässig mit der Partnerfirma ASCOMP der Fortschritt der Arbeit besprochen. In den Besprechungen gefällte Entscheide sind in den Sitzungsprotokollen hinterlegt. Die Sitzungsprotokolle sind im Anhang vorzufinden.

1.5 Qualitätsmassnahmen

1.5.1 Wöchentliche Sitzung

Die bereits im Abschnitt 1.4 erwähnte wöchentliche Fortschrittsbesprechung mit dem Betreuer dient dazu, den aktuellen Fortschritt zu beurteilen und das weitere Vorgehen festzulegen. Die Besprechung steuert dazu bei, das Ziel nicht aus den Augen zu verlieren.

1.5.2 Codereview

Das Projektteam tauscht wöchentlich die Codeerfahrungen aus. Probleme werden besprochen und wenn nötig gemeinsam gelöst. Diese Codereviews sind nicht protokolliert.

1.5.3 Unit-Tests

Für die wichtigsten Komponenten werden Unit-Tests geschrieben. Diese sind im Kapitel 5 vom „Teil 2: Software Dokumentation“ beschrieben.

1.6 Verantwortlichkeiten im Team

Projektbereich / Dokument	Verantwortlicher
Fortlaufende Anpassung des Projektplans	Anita Hollenstein
Protokollierung der Sitzungen	Anita Hollenstein
Zeiterfassung	Anita Hollenstein, Patrice Müller
Q-Massnahmen	Anita Hollenstein, Patrice Müller
Konfigurationsmanagement	Patrice Müller
Entwicklungsprototypen	Anita Hollenstein, Patrice Müller
Architektur	Patrice Müller
Web Front-End	Anita Hollenstein
UNIT Testing	Patrice Müller
Dokumentation	Anita Hollenstein, Patrice Müller

Tabelle 1.2: Verantwortlichkeiten

2 Risikomanagement

Das Ziel ist es immer das Risiko mit dem höchsten Schadenpotenzial zu eliminieren. Die Risiken, welche nicht beeinflusst werden können, wie z.B. Krankheit oder Unfälle sind von dieser Vorgehensweise jedoch ausgeschlossen.

2.1 Risikoanalyse

Bei der Risikoanalyse wurden die Risiken in allgemeine Risiken und in technische Risiken unterteilt. Dabei handelt es sich bei den allgemeinen Risiken um solche, die in jedem Projekt vorkommen können, wie z.B. das ein Projektmitglied krank wird bzw. um Risiken, welche von den externen Partner, wie z.B. ASCOMP, HSR oder Cloudbroker, mit ins Projekt gebracht werden. Bei den implementations-technischen Risiken handelt es sich, wie der Name schon sagt, um Risiken, welche die Implementation betreffen. So könnte es z.B. nicht möglich sein eine Verbindung vom Silverlight Client zu einem Cloud-Backend herzustellen.

2.1.1 Allgemeine Risiken

Risikoname	**Ausfall eines Teammitglieds**	**ID**	**A.1**
Risikostufe	Niedrig	**Max. Schaden**	50 h
Beschreibung	Ein Teammitglied wird von einem Vorfall, wie z.B. Krankheit so stark abgelenkt, dass das Arbeiten verunmöglicht wird.		
Massnahmen	Umfang der Arbeit verkürzen, nacharbeiten so gut es geht.		
Status	Nicht eingetroffen		
Begründung falls eingetroffen	-		

Risikoname	**Ausfall der Infrastruktur**	**ID**	**A.2**
Risikostufe	Sehr niedrig	**Max. Schaden**	50 h
Beschreibung	Die HSR Infrastruktur fällt aus, d.h. kein HPC-Server, keine Arbeitscomputer und kein Internet.		
Massnahmen	- Backup, auf eigenem Computer arbeiten - Es wird ein dezentrales Repository eingerichtet		
Status	Nicht eingetroffen		
Begründung falls eingetroffen	-		

Risikoname	**Projektpartner verspätet sich**	**ID**	**A.3**
Risikostufe	Mittel	**Max. Schaden**	50 h
Beschreibung	Ein Projekt Partner wird mit seinem Teil der Arbeit nicht fertig oder meldet Probleme z.B. bei der Umsetzung.		
Massnahmen	- Mockup - Alternativen		
Status	Eingetroffen		
Begründung falls eingetroffen	1. TransAT wurde nicht wie erwartet vom HPC-Team auf Windows portiert und somit auf dem HPC zum Laufen gebracht. 2. Cloudbroker wurde erst später einsatzbereit als geplant Bei Eintreten des Punktes 1 war auch Cloudbroker noch nicht bereit. Aus diesem Grund wurde vorübergehend ein Mock-up des HPC erstellt, bis dann später Cloudbroker als Alternative verwendet werden konnte.		

Risikoname	**Moving Targets**	**ID**	**A.4**
Risikostufe	Niedrig	**Max. Schaden**	30 h
Beschreibung	Es entstehen unerwartete Zusatzaufgaben, welche zur Unterstützung der Hauptaufgabe benötigt werden.		
Massnahmen	Rücksprache mit dem Betreuer und dem Kunden ob die Zusatzaufgabe aufgeführt oder umgangen wird.		
Status	Eingetroffen		
Begründung falls eingetroffen	Der HPC hatte keine Schnittstelle für den Zugriff von ausserhalb des HSR-Netzwerks. Die Rücksprache ergab, dass die Zusatzaufgabe, also das selbständige Erstellen einer Schnittstelle für den HPC, in Angriff genommen werden soll.		

2.1.2 Risiken der Implementierung

Risikoname	**Kein Zugriff auf ein Cloud-Backend**	**ID**	**I.1**
Risikostufe	Sehr niedrig	**Max. Schaden**	∞
Beschreibung	Weder beim HPC noch bei Cloudbroker kommt ein Zugang zu Stande.		
Massnahmen	Mockup verwenden, jedoch ist die Machbarkeitsstudie fehlgeschlagen.		
Status	Nicht eingetroffen		
Begründung falls eingetroffen	-		

Risikoname	**Schwierigkeiten beim Zugriff auf Cloud-Service**	**ID**	**I.2**
Risikostufe	Niedrig	**Max. Schaden**	30 h
Beschreibung	Der Zugriff auf den Cloud-Service des verwendeten Cloud-Backends hat sich schwieriger gestaltet als gedacht.		
Massnahmen	Deadline setzen, Alternative in der Rückhand behalten.		
Status	Teilweise eingetroffen		
Begründung falls eingetroffen	Es gab zwar Schwierigkeiten, jedoch konnte die Deadline durch zeitintensives Arbeiten eingehalten werden.		

Risikoname	**Datenmenge und Webservice**	**ID**	**I.3**
Risikostufe	Niedrig	**Max. Schaden**	16 h
Beschreibung	Es ist nicht möglich mehrere Gigabyte über einen Webservice zu senden.		
Massnahmen	Übermitteln der Daten per FTP.		
Status	Nicht eingetroffen		
Begründung falls eingetroffen	-		

Risikoname	**Plattformunterschiede**	**ID**	**I.4**
Risikostufe	Mittel	**Max. Schaden**	16 h
Beschreibung	Die Silverlight-Applikation ist aufgrund von Plattform-Unterschieden nicht von jeder Plattform aus zugänglich oder verhält sich unterschiedlich.		
Massnahmen	Fehlerquelle suchen und beheben, falls erfolglos müssen die Plattformen eingeschränkt werden.		
Status	Teilweise eingetroffen		
Begründung falls eingetroffen	Aus unerklärlichen Gründen läuft die Silverlight-Applikation auf dem Firefox-Browser unterschiedlich. Obwohl die zwei Test-Browser dieselbe Version haben, funktioniert die Applikation im einen Browser, in dem anderen jedoch nicht. Da das Problem offensichtlich nicht an der Applikation selbst liegt, wurde das Problem offengelassen.		

3 Entwicklungsumgebung

In diesem Kapitel wird die Entwicklungsumgebung beschrieben. Im Detail wird hier auf Redmine und Git eingegangen, welche das Projektmanagement unterstützten. Es konnte auf eine bestehende Infrastruktur[10] von vergangenen Arbeiten zurückgegriffen werden.

3.1 Redmine

Die Redmine Serverumgebung wurde für das Projekt durch die folgende Plugins erweitert.

- **Scrum-PM:** Diese Plugin ermöglicht das Scrum-Dashboard, Burndown Charts, User-Stories und das Backlog online zu führen.
- **Gitrevision Download Plugin:** Diese Plugin ermöglicht es einem Git-Repository mit dem Browser runter zu laden.

3.2 Git

Git ist ein System zur Verwaltung des Sourcecodes. Es speichert und versioniert den in einem Repository enthaltenen Sourcecode. Somit gehen keine Änderungen verloren und weiter sind diese jederzeit abruf- und kontrollierbar und können zudem rückgängig gemacht werden.

Git wurde wegen seiner Stärke beim Mergen bzw. Branchen gewählt, da dies zentrale Punkte des Entwicklungsprozesses sind.

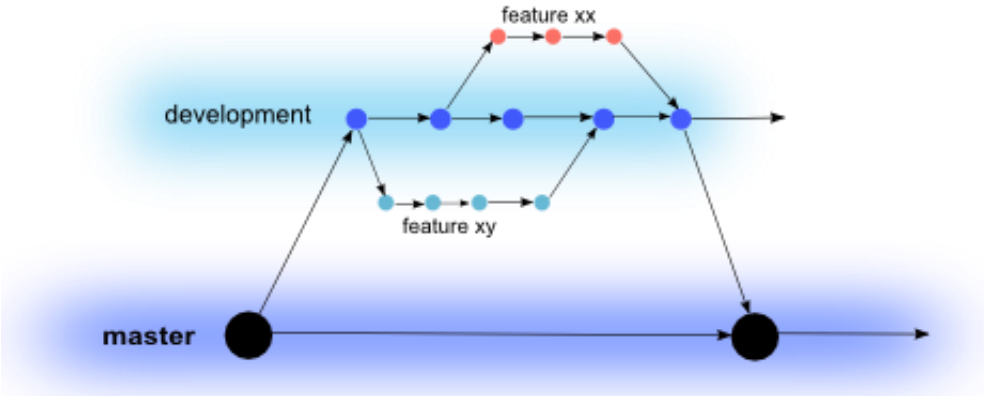

Abbildung 3: Feature-Branches Pattern

Beim Entwicklungsprozess wurde auf das Feature-Branches[11] Pattern von Martin Folwer gesetzt. Das Pattern ist in der Abbildung 3 dargestellt. Daraus folgt diese Aufteilung:

- **master**: Der master Branch wird für voll funktionsfähige Releases genutzt. Er wird am Ende der Arbeit mit dem development Branch gemerged.
- **development**: Im development Branch findet die Entwicklung statt. Dabei wird für jedes neue Feature und jeden neuen Bug ein weiterer Branch erstellt. Dies führt dazu, dass zu einem bestimmten Zeitpunkt niemals mehr als vier Branches existieren. Damit beim Push auf

[10] Ubuntu-Server mit Redmine und eine Git-Dienst
[11] http://martinfowler.com/bliki/FeatureBranch.html

den Server die Revisions-History komplett erhalten bleibt, wird nach folgender Methodik gearbeitet:

```
$ git pull origin development
$ git checkout -b featureXY development
...
   <Arbeit am featureXY>
...
$ git pull origin development
$ git checkout development
$ git merge --no-ff featureXY
...
$ Updating ea1b82a..05e9557
...
$ git branch -d featureXY
...
$ Deleted branch featureXY
...
$ git push origin develop
```

(Müller, 2011)

4 Projektmonitoring

Dieses Kapitel zeigt eine grobe Übersicht der aufgewendeten Arbeitsstunden für die Bachelorarbeit. Ein detaillierter Auszug der Zeiterfassung ist im Anhang beigelegt.

4.1 Ist-Soll-Zeitvergleich

Für die Bachelorarbeit sind pro Student 360 Arbeitsstunden vorgesehen. Über 14 Wochen verteilt entspricht dies einer Arbeitswoche von 51.4 Stunden für beide Studenten zusammen. Die Statistik zeigt, dass im Durchschnitt 60.1 Stunden pro Woche für das Projekt gearbeitet wurde. In der Abbildung 3 ist der Ist-Soll-Vergleich der Stunden pro Woche während der gesamten Projektdauer abgebildet. In der Woche 12 wurde die Implementation abgeschlossen. Da erst spät mit dem Einbinden der Cloudbroker-Cloud gestartet werden konnte, benötigte die Fertigstellung bis zu dem geplanten Implementationsende etwas mehr als die geplante Soll-Zeit. In der zweiten Woche war aus administrativen Gründen der HSR nicht bekannt, ob die Arbeit in der aktuellen Teamsetzung fortgesetzt werden kann. Daraus erfolgte ein kurzer Arbeitsstopp.

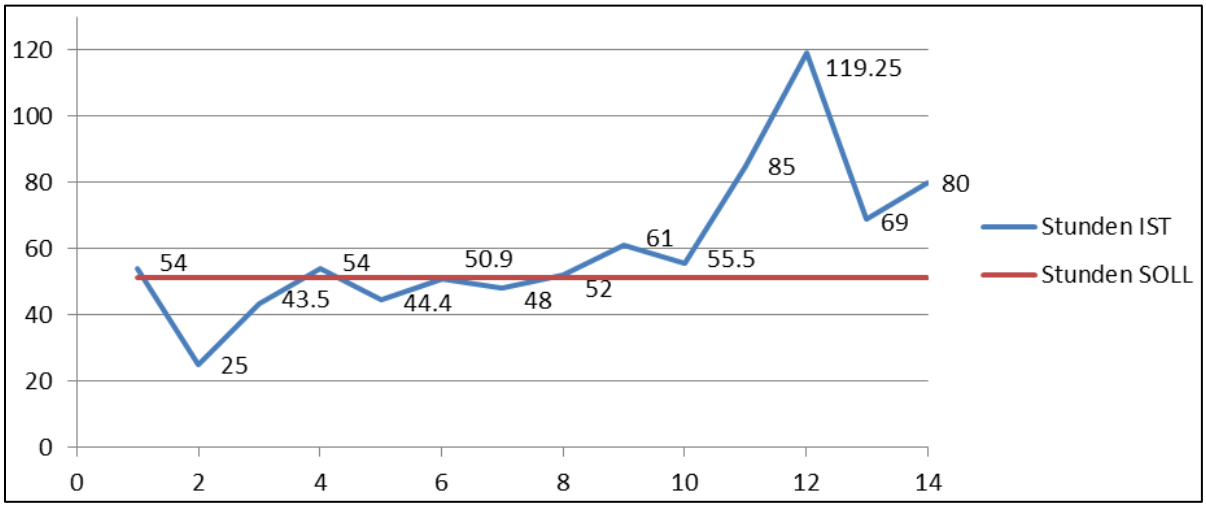

Abbildung 4: Ist-Soll-Zeitvergleich

4.2 Haupttätigkeiten

Die Arbeit wurde in sechs Arbeitsaktivitäten aufgeteilt. Dazu gehören Arbeiten zur Infrastruktur, die Analyse, das Design, das Testing, die Entwicklung und die Dokumentation. In der **Abbildung 5** und der **Abbildung 6** ist zu sehen, dass die Aufteilung sehr ausgeglichen ist. Da Patrice Müller für die Entwicklung des Back-Ends zuständig war, hat er Arbeiten im Bereich Testing durchgeführt. Im Gegenzug war Anita Hollenstein hauptsächlich für die Entwicklung des Front-Ends zuständig. Hier vielen keine Tests an, dafür mussten jedoch im Bereich Design einige Arbeiten durchgeführt werden.

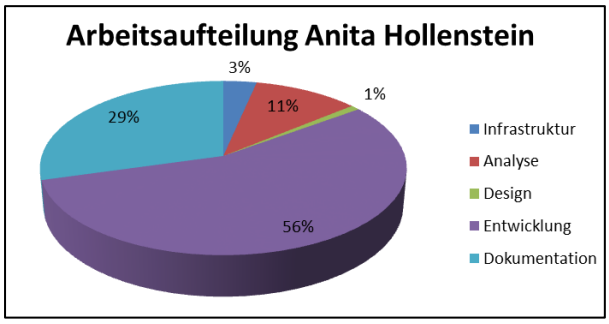

Abbildung 5: Arbeitsaufteilung Anita Hollenstein

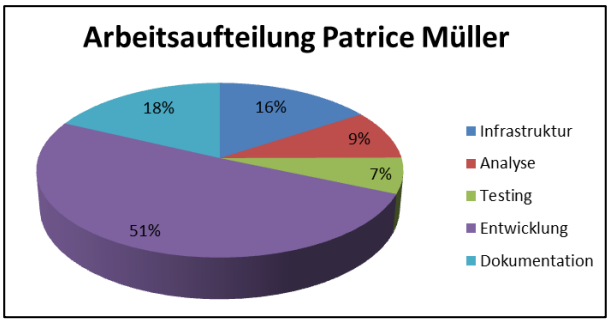

Abbildung 6: Arbeitsaufteilung Patrice Müller

Teil IV

Anhang

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] HUBER, Thomas Claudius: Silverlight 4 Das umfassende Handbuch
Galileo Computing, 2010 – ISBN 978-3836214131

[2] LOWY, Juval Programming WCF Services

O'Reilly, August 30, 2010, ISBN 978-0596805487

Glossar

SSL Secure Sockets Layer, eine Protokoll zur Verschlüsselung der Informationen die über das Netz verschickt werden

REST Representational State Transfer, ist eine Software-Architekturstyle welcher vor allem World Wied Web gebräuchlich ist

XML Extensible Markup Language, ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdaten (wikipedia)

FTP File Transfer Protokoll, ist ein Netzwerkprotokoll welche für den Dateienaustausch optiert ist.

Fortran90 Fortran90 ist eine prozedurale Programmiersprache welche vor allem für numerische Berechnungen eingesetzt wird

P \ Invoke Platform Invocation Service, erlaubt es unmanaged Code in managed Code aufzurufen.

Cloud Computing Ist eine abstrahierte IT-Infrastruktur

Abkürzungsverzeichnis

CRUD Create, Read, Update, Delete: Kurzwort für die Funktionen „Erstellen“, „Lesen“, „Schreiben“ und „Löschen“

GUI Graphical User Interface, auch UI genannt

HPC High Performance Cluster

UI User Interface, auch GUI genannt

WCF Windows Communication Forum

SSL Secure Socket Layer

FTP File Transfer Protocol

REST Representational State Transfer

XML Extensible Markup Language

Aufgabenstellung Bachelorarbeit für Patrice Müller und Anita Hollenstein:

Web UI-Front-End for Fluid Dynamics Cloud

1. Auftraggeber und Betreuer

Diese Bachelorarbeit findet in Zusammenarbeit mit der Firma *Ascomp GmbH* statt.

Ansprechpartner Auftraggeber:

- Dr. Djamel Lakehal, Ascomp GmbH, CEO, lakehal@ascomp.ch
- Daniel Caviezel, Ascomp GmbH, Entwicklungsingenieur, caviezel@ascomp.ch

Betreuer HSR:

- Prof. Dr. Luc Bläser, Institut für Software, lblaeser@hsr.ch

2. Ausgangslage

Die Ascomp GmbH ist ein ETH Physik-Spin-Off, welche im Gebiet der Complex Fluid Dynamics Modellierung und Simulation tätig ist. Um komplexe Fluid Dynamics Simulationen zu rechnen, bedarf es der Ausführung in einem High-Performance-Computing Cluster. Ein solcher Cluster steht insbesondere an der HSR zur Verfügung und könnte zum Beispiel auch im Sinne des Cloud Computings für Fluid Dynamics Simulationen der Firma Ascomp genutzt werden.

Das Durchführen der Simulationen erfordert im Wesentlichen drei Schritte: die Eingabe des Modells mit vielzähligen Parametern (in der Grössenordnung von 1000), das Ausführen der Simulation auf dem HPC-Cluster sowie die Darstellung und Analyse der gerechneten Simulationsresultate. Dieser Prozess findet in der Regel iterativ statt, d. h. die Parameter können jeweils wieder angepasst und die Simulation erneut laufen gelassen werden.

Die Firma Ascomp hat bereits eine UI-Anwendung entwickelt, welche als lokales Programm läuft und die Spezifikation der Fluid Dynamics Simulationen ermöglicht. Die existierende Anwendung unterstützt allerdings keine Cloud-Ausführung, hat noch ungenügende Usability und ist auch in einer etwas älteren Technologie (FLTK) entwickelt.

Die Firma Ascomp ist darauf bestrebt, zukünftig eine komfortable und moderne Benutzerschnittstelle für die Unterstützung des gesamten Simulationsprozesses (Eingabe, Ausführung, Ausgabe) anzubieten. Hierzu sollte ein Web-Interface zur Verfügung gestellt werden, so dass es im Browser von Clienten-Rechner ohne Installation einer lokalen Anwendung flexibel benutzt werden kann.

Das Ziel dieser Bachelorarbeit ist es, eine Machbarkeitsstudie für das „Web UI-Front-End for Fluid Dynamics Cloud" auf der Basis der .NET Silverlight Technologie durchzuführen. Dabei sind insbesondere folgende Aspekte relevant (nicht abschliessend):

1) Unterstützung des Simulations-Workflows mit dem UI-Front-End
2) Datenimport (verschiedene Formate) und Upload via UI
3) Parametereingabe (mit Enabling, Validierung, Konsistenzregeln, Kontextinformationen)
4) Darstellung von 2D und 3D Modellen
5) Schnittstelle zur Cloud (HPC Cluster der HSR, eventuell auch weitere wie „Cloudbroker")
6) Ausführung auf dem Rechencluster
7) Überwachung und Abbruch einer laufenden Clusterrechnung
8) Rückgabe der Resultate und Download via UI
9) Visualisierung der Resultate
10) Flexible Anpassbarkeit des UI an neue oder geänderte Parameter
11) Unterstützung von verschiedenen Versionen der Simulations-Engine
12) Sicherheit bei der Übertragung (wenn nötig)
13) Optional: DB-Verwaltung der Eingaben und Resultate, z.B. mit History

In der Studie ist zuerst eine Grobabklärung aller wesentlichen Funktionalitäten, Schritte und Aspekte für den Gesamt-Workflow nötig und Grobkonzept auszuarbeiten. Die identifizierten Themen sind dabei nach Relevanz und Risiko zu priorisieren. Danach sollen die Themen nach Priorität vertieft analysiert werden und Lösungskonzepte dafür erarbeitet werden.

Aufgrund von Zeitgründen wird nach der Grobanalyse - unter Rücksprache mit dem Betreuer - entschieden, wie umfassend der gesamte Workflow erarbeitet wird oder ob der Fokus hauptsächlich auf die Eingabe gesetzt wird.

Die Machbarkeitsstudie soll als „Proof of Concept" einen exemplarischen Software-Prototyp für das Web UI-Front-End auf der Basis der .NET Silverlight Technologie beinhalten, der die Konzepte der Studie demonstriert. Der Prototyp umfasst zwar die essentiellen Schritte und Funktionalitäten, jedoch mit limitiertem Parameter- und Funktionalitätsumfang. Der Prototyp zeigt insbesondere Lösungen in den Bereichen auf, wo technische Herausforderungen und Risikopunkte identifiziert wurden.

3. Ziele und Aufgabenstellung

Die Aufgabe dieser Arbeit ist es, die Machbarkeitsstudie für das „Web UI-Front-End for Fluid Dynamics Cloud" für die Firma Ascomp durchzuführen.

Folgende spezifische Ziele werden vorgegeben:

- Durchführung der Machbarkeitsstudie für das graphische Web-Front-End zur Fluid Dynamics Simulation in der Cloud. Dies umfasst:
    - Erste Phase: Grobanalyse und Grobkonzept für den gesamten Workflow mit den einzelnen erforderlichen und wünschenswerten Funktionalitäten und Eigenschaften

- Zweite Phase: Vertiefung Analyse von priorisierten Themen und Erarbeitung von detaillierten Lösungskonzepten (z. B. Parametereingabe). Die Priorisierung ist mit den Betreuer abzusprechen.
- Ein Konzept für das UI-Design sowie eine SW-Architektur auf Basis von .NET Silverlight 4 und WPF
- Liste der technischen Risikopunkte mit evaluierten Lösungsmöglichkeiten
- Eine Aufwandschätzung für die vollständige Realisierung

Das Front-End ist auf Basis von .NET Silverlight 4 mit WPF auszulegen und für das Cloud Computing soll primär der Microsoft HPC Cluster der HSR berücksichtigt werden.

- Umfassende Dokumentation der Studie mit allen Analysen, Konzepten und Resultaten.
- Entwicklung eines Software-Prototyps mit beschränktem Umfang, der den Workflow, die wesentlichen Konzepte und die technischen Risikopunkte illustriert. Der Prototyp ist in .NET Silverlight 4 mit WPF zu implementieren. Es können geeignete existierende Bibliotheken oder Lösungen eingesetzt werden.

4. Zur Durchführung

Mit dem HSR-Betreuer und dem Auftraggeber finden in der Regel wöchentliche Besprechungen statt. Zusätzliche Besprechungen sind nach Bedarf durch die Studierenden zu veranlassen. Als technische Kontaktperson der Firma Ascomp steht Herr Daniel Caviezel zur Verfügung.

Alle Besprechungen sind von den Studenten mit einer Traktandenliste vorzubereiten und die Ergebnisse in einem Protokoll zu dokumentieren, das dem Betreuer und dem Auftraggeber per E-Mail zugestellt wird.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. An Meilensteinen gemäss Projektplan sind einzelne Arbeitsresultate in vorläufigen Versionen abzugeben. Über die abgegebenen Arbeitsresultate erhalten die Studierenden ein vorläufiges Feedback. Eine definitive Beurteilung erfolgt auf Grund der am Abgabetermin abgelieferten Dokumentation.

5. Dokumentation

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen (siehe https://www.hsr.ch/Allgemeine-Infos-Diplom-Bach.4418.0.html?&L=0). Die zu erstellenden Dokumente sind im Projektplan festzuhalten. Alle Dokumente sind nachzuführen, d.h. sie sollten den Stand der Arbeit bei der Abgabe in konsistenter Form dokumentieren. Die Dokumentation ist vollständig auf CD/DVD in 3 Exemplaren abzugeben. Auf Wunsch ist für den Auftraggeber eine gedruckte Version zu erstellen.

6. Video-Demo

Für diese Arbeit ist eine kurze Video-Demo zu erstellen. Dieses Video sollte nicht mehr als 2 Minuten lang sein. Das Video sollte einen Intro-Screen mit HSR-Logo, Titel der Arbeit, Namen der Studenten, Namen des Betreuers/Dozenten enthalten (5 sec). Im Hauptteil des Videos sollte das Problem kurz beschrieben werden (kann ein Standbild/Folie sein) und dann eine kurze Demo entsprechend einem wichtigen (realistischen) Nutzungsszenario zeigen (Person und Screen). Im Abspann sollte der Intro-Screen wieder eingeblendet werden (3 sec). Material für die Video-Erstellung wird von der HSR gestellt (Multimediabestellungen).

7. Termine

Siehe auch Terminplan auf https://www.hsr.ch/Termine-Diplom-Bachelor-und.5142.0.html?&L=0

19.09.11	Beginn der Bachelorarbeit, Ausgabe der Aufgabenstellung durch die Betreuer.
November 11	Fotoshooting. Genauere Angaben erteilt die Kommunikationsstelle rechtzeitig.
15.12.11	Die Studierenden geben den Abstract für die Diplomarbeitsbroschüre zur Kontrolle an ihren Betreuer/Examinator frei. Die Studierenden erhalten vorgängig vom Studiengangsekretariat die Aufforderung mit den Zugangsdaten zur Online-Erfassung des Abstracts für die Broschüre. Die Studierenden senden per Email das A0-Poster zur Prüfung an ihren Examinator/Betreuer. Vorlagen sowie eine ausführliche Anleitung betreffend Dokumentation stehen unter den allgemeinen Infos Diplom-, Bachelor- und Studienarbeiten zur Verfügung.
20.12.11	Der Betreuer/Examinator gibt das Dokument mit dem korrekten und vollständigen Abstract der Broschüre zur Weiterverarbeitung an das Studiengangsekretariat frei.
23.12.11	Abgabe des Berichtes an den Betreuer bis 12.00 Uhr. Fertigstellung des A0-Posters bis 12.00 Uhr.
03.01.12 - 03.02.12	Mündliche BA-Prüfung
02.03.12	Nachmittag, Diplomübergabe und Ausstellung Bachelorarbeiten

8. Beurteilung

Eine erfolgreiche Bachelorarbeit zählt 12 ECTS-Punkte pro Studierenden. Für 1 ECTS Punkt ist eine Arbeitsleistung von ca. 25 bis 30 Stunden budgetiert. Für die Modulbeschreibung der Bachelorarbeit siehe auch https://unterricht.hsr.ch/staticWeb/allModules/19419_M_BAI.html.

Für die Beurteilung sind die HSR-Betreuer verantwortlich.

Gesichtspunkt	Gewicht
1. Organisation, Durchführung	1/6
2. Berichte (Abstract, Mgmt Summary, technischer u. persönliche Berichte) sowie Gliederung, Darstellung, Sprache der gesamten Dokumentation	1/6

3. Inhalt *)	3/6
4. Mündliche Prüfung zur Bachelorarbeit	1/6

*) Die Unterteilung und Gewichtung von 3. Inhalt wird im Laufe dieser Arbeit mit den Studierenden festgelegt.

Im Übrigen gelten die Bestimmungen der Abteilung Informatik für Bachelorarbeiten.

Rapperswil, den 14. September 2011

Der verantwortliche Dozent

Prof. Dr. Luc Bläser
Institut für Software
Hochschule für Technik Rapperswil

Vereinbarung

1. Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Bachelorarbeit **«WebUI Front-End for Fluid Dynamics Cloud Computing»** von **Anita Hollenstein** und **Patrice Müller** unter der Betreuung von Prof. Dr. Luc Bläser geregelt.

2. Urheberrecht

Die Urheberrechte stehen den Studenten zu.

3. Verwendung

Die Ergebnisse der Arbeit dürfen sowohl von den Studenten, von der HSR sowie von der ASCOMP GmbH, Zürich, nach Abschluss der Arbeit verwendet und weiterentwickelt werden. Die Verwendung und/oder Weiterentwicklung bedarf keiner Nachfrage bei den Urhebern oder den Nutzern. Copyright-Bemerkungen dürfen nicht entfernt werden.

Rapperswil, den 21.12.2011 .....

Die Studentin Anita Hollenstein

Rapperswil, den 21.12.2011 .....

Der Student Patrice Müller

Rapperswil, den 21.12.11 .....

Der Betreuer der Bachelorarbeit

Management Summary

Ausgangslage

Das Anliegen von ASCOMP
Die ASCOMP GmbH ist mit dem Fluiddynamik Berechnungsprogramm TransAT international tätig. Das Ziel dieser Arbeit ist eine Machbarkeitsstudie, die zeigt, dass es prinzipiell möglich ist, TransAT in eine Rechner-Cloud auszulagern und über eine webbasierte grafische Benutzeroberfläche auf einfache Weise zugänglich zu machen. Der Vorteil an diesem Ansatz ist das der Endbenutzer keine Installation machen muss, sondern einfach den Webbrowser startet und dann gleich loslegen kann. Die grafische Oberfläche der Applikation soll den Endnutzer bei der Orientierung in seinem Arbeitsfluss unterstützen.

Konkurrenz
Bis jetzt gibt es noch keine Software im Bereich der Fluid Dynamik Simulation, welche die Simulationsberechnungen in die Cloud auslagert und zugleich einen Silverlight-Client oder eine ähnliche „No-installation"-Applikation als grafische Oberfläche zur Verfügung stellt.

Warum wurde das Projekt bearbeitet
Da uns die Kombination der Silverlight-Technologie mit der Thematik Cloud-Computing sehr interessiert hat.

Ziele
Machbarkeitsstudie zum WebUI Front-End, die mit Hilfe eines Prototyps die Machbarkeit, dass das Silverlight Front-End an das Cloud-Backend abgebunden werden kann sowie das mit Silverlight eine ansprechende und funktionierende Benutzeroberfläche erstellt werden kann.

Vorgehen

Die Teilschritte
1. Verbindung zu einem Cloud-Backend, dem HPC der HSR, herstellen. Weiter wurde parallel dazu mit der Entwicklung der grafischen Oberfläche und der Logik des Silverlight-Clients angefangen.
2. Erster Prototyp, der die Projektdateien in die Cloud laden kann.
3. Zweiter Prototyp: Die Dateien werden vom Client auf die Cloud geladen und nach der Fertigstellung der Berechnung die Lösungsdatei heruntergeladen.
4. Anbindung an das zweite Cloud-Backend, Cloudbroker
5. Zusammenführung der einzelnen Prototypen zum finalen Prototyp.

Risikoelimination
Es wurde stets bestrebt das nächst grösste Risiko durch Abklärungen und oder Implementierung zu eliminieren.

Beteiligte
Die Arbeit wurde von den Studenten Anita Hollenstein und Patrice Müller durchgeführt. Als Betreuer stand ihnen Prof. Dr. Luc Bläser zu Seite.

Die ASCOMP, mit welcher ca. alle drei Wochen ein Meeting abgehalten wurde, war der Auftraggeber dieser Arbeit.

Als Cloud-Partner standen das „Institut für Energietechnik (IET) & Microsoft Technical Computing Innovation Center“ der HSR und die Firma Cloudbroker zur Seite. Das Institut begleitete die Arbeit bis und mit zum Prototyp 3, danach wurde zur Firma Cloudbroker gewechselt, da wider Erwarten der TransAT Solver nicht rechtzeitig von anderen Entwicklern auf den HPC portiert worden ist.

Erreichte Ziele

Das Hauptziel, die Machbarkeit des Projektes zu zeigen, sowie alle zuvor erwähnten Ziele wurden erfüllt. Das heisst, das Front-End wurde erfolgreich an die Schnittstelle zu einem Cloud-Backend angeschlossen. Das WebUI Front-End stellt dem Benutzer folgende Möglichkeiten zur Verfügung:

- Übersicht und Verwaltung der Simulationsprojekte
- Reduzierte Parameter-Editierung mit grafischer und funktionaler Unterstützung des Arbeitsflusses
- Upload von Projekten auf eine Cloudbroker-Cloud zur Berechnung
- Download der berechneten Lösung mit der Möglichkeit, diese lokal darzustellen
- Schnittstelle zur Cloud-Infrastruktur von Cloudbroker
- Schnittstelle zum HPC der HSR mit simuliertem HPC Berechnungsablauf

Was wurde gemacht

Der Silverlight-Client wurde komplett von den Studenten entwickelt. In der Abbildung 7 ist ein Screen Snapshot der Erstellten Benutzeroberfläche abgebildet. Der HPC war erst seit kurzer Zeit in Betrieb und noch nicht darauf ausgerichtet, von ausserhalb des HSR-Netzwerks benutzt zu werden. Zur Überbrückung dieser Einschränkung musste daher zunächst ein Zwischen-Service entwickelt werden. In der zweiten Hälfte der Bachelorarbeit gab Cloudbroker, der zweite Cloud-Partner dieser Arbeit, die Fertigstellung der Schnittstelle zu seiner Cloud-Infrastruktur bekannt. Schliesslich ist es gelungen, Cloudbroker funktionsfähig an den Prototypen anzubinden und für die TransAT-Berechnung der Simulationsprojekte aus dem WebUI Front-End einzusetzen.

Abbildung 7: Screen-Snapshot der Benutzeroberläche

Was nehmen die Studenten aus dem Projekt mit

Die Studenten durften ihre ersten Erfahrungen in einem Projekt sammeln wo mehr als nur eine Firma daran beteiligt war, mit alle den angenehmen und etwas weniger angenehmen Seiten. Es wurde

nämlich erkannt, dass je mehr Parteien am einem Projekt beteiligt sind, desto schwieriger wird es einen Überblick über das Projekt zu behalten. Weiter ist man von Leistungen der anderen Abhängig. Wenn diese verspätet sein, dann wir automatisch auch das eigene Projekt verspätet.

Ausblick

Für eine Weiterentwicklung empfiehlt es sich die folgenden Punkte anzuschauen.

- Projekt-, Benutzer- und Lizenz-Verwaltung
- UI-Management, das UI soll einfach anpassbar sein
- Konfigurations-Management des Berechnungsauftrages. Dem Endbenutzer soll es ermöglicht werden, z.B. die Startzeit seines Auftrags bestimmen zu können.
- 3D Rendering in der Silverlight Applikation (Silverlight 5), es soll dem Endbenutzer ermöglicht werden, im Menu „Mesh“ seine Objekte in 3D zu sehen.

Konfigurationsmöglichkeiten des Windows Service vom Entwicklungsprototyp 1

Start-Parameter	XML	Standartwert	Name	Beschreibung
-ru	remoteUser	ascomp	Remote User	Benutzername des lokalen Benutzer auf dem HPC-Proxy
-rp	remotePW	gnome32bit#	Passwort	Passwort des Benutzer auf den HPC-Proxy
-rd	remoteDomain	sinv-56020	Remote Domain	Domain des HPC-Proxy‘s
-rrp	remoteRootPath	\\152.96.56.20\TransATWebUI	Remote Root Path	Der Root-Pfad in welchem sich die Ordner Job und Solution befinden.
-jsp	jobPath	\Job	Job Source Path	Das sind die Job-Dateien die vom Benutzer auf den HPC-Proxy geladen wurden und jetzt warten vom HPC geholt zu werden.
-jdp	jobDestinationPath	C:\Users\Public\Job	Job Destination Path	Hierhin werden die Job-Dateien kopiert um dann später berechnet zu werden. (benutzer_projektname_job.zip)
-ssp	solutionPath	\Solution		Hier hin werden die Fertigen Berechneten und gepackten Daten verschoben um vom Benutzer runtergeladen zu werden.
-sdp	solutionDestinationPath	C:\Users\Public\Solution		Hier werden die Solution-Dateien auf dem HPC gespeichert bis sie auf den HPC-Proxy hoch geladen werden.
-c		false	CleanUp on Start	Lädt erst alle evtl. vorhandenen alle Solution-Dateien auf den HPC-Proxy bzw. holt alle Job-Dateien vom HPC-Proxy und fängt erst dann an zu schauen ob es neue gibt.

XML-Datei (IIS_Filewatcher.exe.config)

Die XML-Datei Namens „IIS_Filewatcher.exe.config“ finden man im gleichen Verzeichnis wie den Service (IIS_Filewatcher.exe) selbst.

```
<setting name="remoteDomain" serializeAs="String">

       <value>sinv-56020</value>

</setting>
```

CloudBroker

CloudBroker Platform
REST API Usage Manual

Version 1.0

CloudBroker GmbH
Technoparkstrasse 1, CH-8005 Zürich, Switzerland
Phone: +41 44 633 79 34
Email: info@cloudbroker.com
Web: http://www.cloudbroker.com

Introduction

Our platform provides users with a reliable and simple REST-based web service API, which allows you to do almost all the actions that are available from the web browser interface. The platform API provides responses in suitable XML format, which can be easily used further in your software.

All requests in this manual were made using the curl tool. The general request format is:

```
curl -k -u "email:password" requested_address
```

However, it should be possible to do the same requests in any programming language that is able to issue HTTP requests.

We will use here the creation and execution of a job as example. However, all other objects and actions in the platform should work in a similar way.

Note that “link_to_server” used in URLs should be replaced with “https://labs.cloudbroker.com”.

Table of Contents

List Softwares

Purpose

This operation lists all available softwares. Softwares are application program packages that can be used on the available cloud resource. We use the pseudo-plural "softwares" here to denote the difference between all application program packages ("softwares") and a single one ("software"). You will need a software ID or name during job creation.

URL: link_to_server/softwares.xml

Request Type: GET

Request Parameters

None

Response Parameters

Parameters	Description
description	Software description
documentation-link	Link to the software documentation
id	Software ID
name	Software name
organization-id	Software owner ID
product	Product name
status	Software status
use-nfs	Specifies if the software uses a shared file system Type: boolean
version	Software version
website	Software website

Example Request

```
curl -k -u email:password link_to_server/softwares.xml
```

Example Response

```
<softwares>
 <software>
  <description>X! Tandem is software that can match tandem mass spectra with peptide sequences, in a process that has come to be known as protein identification.</description>
  <documentation-link>http://www.thegpm.org/TANDEM/</documentation-link>
  <id>70bdc83f-f224-4456-acbd-d29a6759b3ae</id>
  <name>X! Tandem TORNADO (2010.01.01.4)</name>
  <organization-id>eda2172e-cc7d-478c-8215-214ad3a974c7</organization-id>
```

```
    <product>X! Tandem</product>
    <status>active</status>
    <use-nfs type="boolean">false</use-nfs>
    <version>TORNADO (2010.01.01.4)</version>
    <website>http://www.thegpm.org/TANDEM/</website>
  </software>
</softwares>
```

List Executables

Purpose

This operation lists all available executables. Executables are binaries within the given software package. You will need an executable ID or name during job creation.

URL: link_to_server/executables.xml

Request Type: GET

Request Parameters

None

Response Parameters

Parameters	Description
active	Defines whether the executable can be used in a job Type: boolean
binary	Executable binary
description	Executable description
id	Executable ID
name	Executable name
software-id	Software ID the executable belongs to

Example Request

```
curl -k -u email:password link_to_server/executables.xml
```

Example Response

```
<executables>
 <executable>
  <active type="boolean">true</active>
  <binary>trjconv</binary>
  <description>Gromacs utility</description>
  <id>f65e1346-de3a-4277-9a39-13ce21f57145</id>
  <name>Gromacs 4.0.7 trjconv</name>
  <software-id>7782438e-d14e-4bde-b258-9d0ba2732a96</software-id>
 </executable>
</executables>
```

List Resources

Purpose

This operation lists all available resources. A resource is a cloud infrastructure within which instances can be launched. You will need a resource ID or name during job creation.

URL: link_to_server/resources.xml

Request Type: GET

Request Parameters

None

Response Parameters

Parameters	Description
active	Defines whether the resource is active Type: boolean
description	Resource description
id	Resource ID
maximum-nodes	Maximum number of instances that can be used within the resource Type: integer
name	Resource name
website	Resource website

Example Request

```
curl -k -u email:password link_to_server/resources.xml
```

Example Response

```
<resources>
 <resource>
  <active type="boolean">true</active>
  <description>Amazon Elastic Compute Cloud</description>
  <id>1f898f26-e016-4e12-8ae5-5b6bb311afb3</id>
  <maximum-nodes type="integer">20</maximum-nodes>
  <name>Amazon EC2</name>
  <website>http://aws.amazon.com</website>
 </resource>
</resources>
```

List Regions

Purpose

This operation lists all available regions. Regions are geographical zones or data centers within a cloud resource. You will need a region ID or name during job creation.

URL: link_to_server/regions.xml

Request Type: GET

Request Parameters

None

Response Parameters

Parameters	Description
ec2-clusters	Specifies whether it is possible to use Amazon EC2 Cluster Compute or Cluster GPU instances in this region Type: boolean
id	Region ID
maximum-nodes	Defines the maximum number of instances that can be used within the region Type: integer
name	Region name
resource-id	ID of the resource, within which the region is located

Example Request

```
curl -k -u email:password link_to_server/regions.xml
```

Example Response

```
<region>
  <ec2-clusters type="boolean">false</ec2-clusters>
  <id>bc26f58e-fd72-4eef-a416-c6cb19ce0734</id>
  <maximum-nodes type="integer" nil="true"></maximum-nodes>
  <name>us-west-1</name>
  <resource-id>1f898f26-e016-4e12-8ae5-5b6bb311afb3</resource-id>
</region>
```

List Instance Types

Purpose

This operation lists all available instance types. Instance types are types of virtual or physical machines within a cloud resource. You will need an instance type ID or name during job creation.

URL: link_to_server/instance_types.xml

Request Type: GET

Request Parameters

None

Response Parameters

Parameters	Description
cpus	Specifies the number of CPUs an instance has Type: integer
description	Instance type description
ebs-only	Defines whether only elastic block store-based instances can be launched from this instance type Type: boolean
ec2-clusters	Specifies whether this instance type issues Amazon EC2 Cluster Compute or Cluster GPU instances Type: boolean
id	Instance type ID
instance-storage	Specifies the storage capacity of an instance in GB Type: integer
maximum-nodes	Maximum number of instances that can be launched for this instance type Type: integer
memory	Specifies the memory capacity of an instance in GB Type: decimal
name	Instance type name
resource-id	ID of the resource an instance is launched in
root-storage	Specifies the root storage capacity of an instance in GB

	Type: integer

Example Request

```
curl -k -u email:password link_to_server/instance_types.xml
```

Example Response

```
<instance-types>
  <instance-type>
    <cpus type="integer">1</cpus>
    <description>Amazon EC2 m1.small instance type</description>
    <ebs-only type="boolean" nil="true"></ebs-only>
    <ec2-clusters type="boolean">false</ec2-clusters>
    <id>129f0d6f-d0fa-43d9-9b73-2c765a14c648</id>
    <instance-storage type="integer" nil="true"></instance-storage>
    <maximum-nodes type="integer" nil="true"></maximum-nodes>
    <memory type="decimal">1.7</memory>
    <name>Amazon EC2 m1.small i386</name>
    <resource-id>1f898f26-e016-4e12-8ae5-5b6bb311afb3</resource-id>
    <root-storage type="integer" nil="true"></root-storage>
  </instance-type>
</instance-types>
```

List Jobs

Purpose

This operation lists all available jobs. Jobs are computational tasks that run a software executable on a cloud resource. You will need a software ID or name, executable ID or name, resource ID or name, region ID or name and instance type ID or name during job creation.

URL: link_to_server/jobs.xml

Request Type: GET

Request Parameters

None

Response Parameters

Parameters	Description
access-id	ID of the access used with this job
archive-output	Specifies whether the output datafiles should be packaged and compressed Type: boolean
argument-string	Specifies an argument string for the executable
description	Job description
executable-id	ID of the executable used with the job
id	Job ID
instance-type-id	ID of the instance type used with the job
job-outcome-id	ID of the job outcome status
license-id	ID of the license used with this job
name	Job name
nodes	Specifies the number of instances used to execute this job Type: integer
previous-job-id	ID of the job that is executed previously to this job in a pipeline
reuse-inputs-from-job-id	Specifies the job ID to reuse the input datafiles from
start-immediately	Defines whether a job shall be started immediately or should wait until there are free instances to reuse

status	Defines the job execution status
stderr-file-name	Defines the standard error filename
stdout-file-name	Defines the standard output filename
stop-reason	Defines the reason for interruption of job execution
storage-region-id	Defines the ID of the region where the job datafiles are stored
use-ebs-ami	Specifies whether an elastic block store-backed Amazon machine image should be used Type: boolean
used-core-runtime	Defines the used time length of job execution in conversion to one instance core in hours Type: integer
used-runtime	Defines a used time length of job execution in hours Type: integer
used-storage	Defines the storage space used for the job datafiles in bytes Type: integer
user-id	ID of the user submitting the job

Example Request

```
curl -k -u email:password link_to_server/jobs.xml
```

Example Response

```
<jobs>
  <job>
    <access-id>019f3fd3-6e15-4405-8ba4-9a9a3193f060</access-id>
    <archive-output type="boolean">false</archive-output>
    <argument-string></argument-string>
    <description></description>
    <executable-id>3530f805-8576-4a4a-add6-72a5d896e2d3</executable-id>
    <id>a087e4d4-cf1b-4956-8c8a-6fab4d34c2e8</id>
    <instance-type-id>711f4951-3c45-4ce8-86a2-a27cd653d912</instance-type-id>
    <job-outcome-id>23305791-c3d0-4281-99b0-60e25daa7033</job-outcome-id>
    <license-id>31bfb9fb-c65d-412f-9f22-2e7fe9922cd4</license-id>
    <name>Gromacs Job</name>
    <nodes type="integer">1</nodes>
    <previous-job-id></previous-job-id>
    <reuse-inputs-from-job-id nil="true"></reuse-inputs-from-job-id>
    <start-immediately type="boolean">true</start-immediately>
    <status>completed</status>
    <stderr-file-name>job.err</stderr-file-name>
    <stdout-file-name>job.out</stdout-file-name>
    <stop-reason nil="true"></stop-reason>
    <storage-region-id>2c4a5365-95cd-449f-b40d-36da48c93118</storage-region-id>
    <use-ebs-ami type="boolean">true</use-ebs-ami>
    <used-core-runtime type="integer">2</used-core-runtime>
    <used-runtime type="integer">1</used-runtime>
    <used-storage type="integer">0</used-storage>
    <user-id>fa87673a-b971-4ef2-b1a6-df38b13816b6</user-id>
```

```
    </job>
</jobs>
```

List Datafiles

Purpose

This operation lists all available datafiles. Datafiles can be input files for job submission, as well as output files containing job execution results. You will need a datafile ID or name during job creation.

URL: link_to_server/data_files.xml

Request Type: GET

Request Parameters

None

Response Parameters

Parameters	Description
archive	Specifies whether a datafile is packed and compressed Type: boolean
data-content-type	Specifies the data content type
data-file-name	Datafile name
data-file-size	Datafile size in bytes Type: integer
data-type-id	Datatype ID (defines whether the file is input or output)
data-updated-at	Defines the datafile update date and time Type: datetime
description	Datafile description
id	Datafile ID
job-id	ID of the job the datafile belongs to

Example Request

```
curl -k -u email:password link_to_server/data_files.xml
```

Example Response

```
<data-files>
 <data-file>
  <archive type="boolean">false</archive>
  <data-content-type nil="true"></data-content-type>
  <data-file-name>output.tgz</data-file-name>
  <data-file-size type="integer">70389</data-file-size>
  <data-type-id>ed408ca0-92f4-4227-93db-2c2fb02e244c</data-type-id>
  <data-updated-at type="datetime" nil="true"></data-updated-at>
```

```
  <description nil="true"></description>
  <id>8c890955-3b86-4429-9b62-34a1d707246a</id>
  <job-id>2db1432b-bdc8-468c-b67d-17b54c062c27</job-id>
 </data-file>
</data-files>
```

Create Job

Purpose

This operation allows you to create a job through the API.

URL: link_to_server/jobs.xml

Request Type: POST

Request Parameters

Parameter	Description	Required
name	Job name	yes
software_id	Software ID	at least one of
software_name	Software name	
executable_id	Executable ID	at least one of
executable_name	Executable name	
resource_id	Resource ID	at least one of
resource_name	Resource name	
region_id	Region ID	at least one of
region_name	Region name	
instance_type_id	Instance type ID	at least one of
instance_type_name	Instance type name	
use_ebs_ami	Specifies whether an elastic block store-backed Amazon machine image should be used Type: boolean	no (default: false)
nodes	Specifies the number of instances to be used to execute this job Type: integer	no (default: 1)
argument_string	Specifies the argument string for the executable	no
stdout_file_name	Defines the standard output name	no (default: job.out)
stderr_file_name	Defines the standard error name	no (default: job.err)

archive_output	Specifies whether the output datafiles should be packaged and compressed Type: boolean	no (default: false)
start_immediately	Defines whether a job shall be started immediately or wait for free instances to appear. If unsure, set this to true.	no (default: true)
description	Job description	no

In case the job has been successfully created, the following response parameters will be returned:

Response Parameters

Parameters	Description
access-id	ID of the access used with this job
archive-output	Specifies whether the output datafiles should be packaged and compressed Type: boolean
argument-string	Specifies an argument string for the executable
description	Job description
executable-id	ID of the executable used with the job
id	Job ID
instance-type-id	ID of the instance type used with the job
job-outcome-id	Outcome of a job ('unknown' when created)
license-id	ID of the license used with this job
name	Job name
nodes	Specifies the number of instances used to execute this job Type: integer
previous-job-id	ID of the previous job executed in a pipeline
reuse-inputs-from-job-id	Specifies the job ID to reuse the input datafiles from
start-immediately	Defines whether a job shall be started immediately or if it shall

	wait for instances to free up
status	Job execution status
stderr-file-name	Defines the standard error name
stdout-file-name	Defines the standard output name
stop-reason	Defines the reason for interruption of job execution
storage-region-id	Defines the ID of the region where the job datafiles are stored
use-ebs-ami	Specifies whether an elastic block store-backed Amazon machine image is used Type: boolean
used-core-runtime	Defines the used core runtime, i.e. the number of hours the job ran multiplied by the number of cores multiplied by the number of nodes Type: integer
used-runtime	Defines the used runtime, i.e. the number of hours the job ran multiplied by the number of nodes Type: integer
used-storage	Defines the storage space used for job execution in bytes Type: integer
user-id	ID of the user submitting the job

Example Request

```
curl -k -u email:password
-H "Content-Type: application/xml"
-i -X POST
-d "<job>
  <name>JobCreatedWithAPI</name>
  <software_id>9be66892-5691-4855-93c8-ea422bf034ba</software_id>
  <executable_id>24142f07-891a-4efa-ad8e-7da1b65eef5c</executable_id>
  <resource_id>96384fb2-9090-4efa-8f79-8e0c538b7941</resource_id>
  <region_id>1feb4f64-324b-467a-99fc-10d8a60dab71</region_id>
  <instance_type_id>d147e81f-8b00-4d70-ac92-a7400bb46308</instance_type_id>
  <use_ebs_ami>false</use_ebs_ami>
  <previous_job_id></previous_job_id>
  <nodes>1</nodes>
  <argument_string>-p pkf</argument_string>
  <stdout_file_name></stdout_file_name>
  <stderr_file_name></stderr_file_name>
  <archive_output>false</archive_output>
  <start_immediately>true</start_immediately>
  <description>This job was created through API</description>
</job>"
link_to_server/jobs.xml
```

Example Response

```
HTTP/1.1 201 Created
Connection: close
Date: Tue, 26 Oct 2010 13:48:15 GMT
X-Runtime: 2268
Location: link_to_server/jobs/99985d29-b137-4fd7-aaa5-1ef1ed0a5b6a
Content-Type: application/xml; charset=utf-8
Cache-Control: no-cache
Set-Cookie: _CloudBroker_session=d0d8ad2c8d747d6d60caa10c3c22db6d; path=/; HttpOnly
Content-Length: 1274


<?xml version="1.0" encoding="UTF-8"?>
<job>
  <access-id>420bcb4d-9add-4661-bb8e-8a819056e738</access-id>
  <archive-output type="boolean">false</archive-output>
  <argument-string>-p pkf</argument-string>
  <description>This job was created through API</description>
  <executable-id>24142f07-891a-4efa-ad8e-7da1b65eef5c</executable-id>
  <id>99985d29-b137-4fd7-aaa5-1ef1ed0a5b6a</id>
  <instance-type-id>d147e81f-8b00-4d70-ac92-a7400bb46308</instance-type-id>
  <job-outcome-id>ae8003ea-4454-4e4c-876d-1ccb469d4fa6</job-outcome-id>
  <license-id>4a1975e4-cfa7-4b5d-a582-50cc27ff746e</license-id>
  <name>JobCreatedWithAPI</name>
  <nodes type="integer">1</nodes>
  <previous-job-id nil="true"></previous-job-id>
  <start-immediately type="boolean">true</start-immediately>
  <status>created</status>
  <stderr-file-name>job.err</stderr-file-name>
  <stdout-file-name>job.out</stdout-file-name>
  <stop-reason nil="true"></stop-reason>
  <use-ebs-ami type="boolean">false</use-ebs-ami>
  <used-core-runtime type="integer">0</used-core-runtime>
  <used-runtime type="integer">0</used-runtime>
  <used-storage type="integer">0</used-storage>
  <user-id>879fb0f1-a882-49a4-82d4-bbff6a7a6b3d</user-id>
</job>
```

If there are errors during job creation, there will be a list of them in the response of the following format:

Example Error

```
HTTP/1.1 422
Connection: close
Date: Wed, 27 Oct 2010 08:45:26 GMT
X-Runtime: 3688
Content-Type: application/xml; charset=utf-8
Cache-Control: no-cache
Set-Cookie: _CloudBroker_session=f502a22787bf2f4d11af69d3801d0c73; path=/; HttpOnly
Content-Length: 115

<?xml version="1.0" encoding="UTF-8"?>
<errors>
  <error>You already have the job with such name</error>
</errors>
```

Submit Job

Purpose

This operation allows you to submit a job through the API.

URL: link_to_server/jobs/job_id/submit.xml

Request Type: PUT

Request Parameters

None

Response Parameters

In case the job has been successfully submitted, an OK response will be returned.

Example Request

```
curl -k -u "email:password"
-X PUT
link_to_server/jobs/e8a448fc-22c9-2356-12344cb52e2e/submit.xml
```

Example Error

If there is a "Length required" error during submission, just add an empty body to the request:

```
curl -k -u "email:password"
-X PUT
link_to_server/jobs/e8a448fc-22c9-4269-822b-3297cb58e2ec/submit.xml -d ''
```

Create Datafile

Purpose

This operation allows you to create a datafile through the API.

URL: link_to_server/data_files.xml

Request Type: POST

In order to create a datafile API, the following request parameters should be sent:

Request Parameters

Parameter	Description	Required
data	Path to the file that should be uploaded	yes
job_id	ID of the job the datafile belongs to	yes
archive	Specifies whether the datafile should be packaged and compressed Type: boolean	no
data_type_id	Datafile data type ID	yes
description	Datafile description	no

In case the datafile has been successfully created, the following response parameters will be returned:

Response Parameters

Parameters	Description
archive	Specifies whether the datafile should be packaged and compressed Type: boolean
data-content-type	Specifies the data content type
data-file-name	Datafile name
data-file-size	Datafile size Type: integer
data-type-id	Datafile type ID
data-updated-at	Defines datafile update date and time

	Type: datetime
description	Datafile description
id	Datafile ID
job-id	ID of the job the datafile belongs to

Example Request

```
curl -k -u "email:password"
-F data=@d:\CloudBroker\cbp.txt
-F job_id=99985d29-b137-4fd7-aaa5-1ef1ed0a5b6a
-F archive=false
-F data_type_id=be828165-d483-4218-8e9b-5981f7637b83
-F description=API
link_to_server/data_files.xml
```

Example Response

```
<data-file>
  <archive type="boolean">false</archive>
  <data-content-type nil="true"></data-content-type>
  <data-file-name>output.tgz</data-file-name>
  <data-file-size type="integer">70389</data-file-size>
  <data-type-id>ed408ca0-92f4-4227-93db-2c2fb02e244c</data-type-id>
  <data-updated-at type="datetime" nil="true"></data-updated-at>
  <description nil="true"></description>
  <id>8c890955-3b86-4429-9b62-34a1d707246a</id>
  <job-id>2db1432b-bdc8-468c-b67d-17b54c062c27</job-id>
</data-file>
```

Download Datafile

Purpose

This operation allows you to download a datafile through the API.

URL: link_to_server/data_files/data_files_id/download.xml

Request Type: PUT

Request Parameters

None

Response Parameters

None

A link to the datafile will be returned to you.

Example Request

```
curl -k -u "email:password"
-X PUT
link_to_server/data_files/62cd7ab1-b05e-c628e-e1dd447462685/download.xml
```

Example Response

```
HTTP/1.1 302 Moved Temporarily
Connection: close
Date: Wed, 27 Oct 2010 15:39:53 GMT
X-Runtime: 704
Location: link_to_datafile
Content-Type: text/html; charset=utf-8
Cache-Control: no-cache
Set-Cookie: _CloudBroker_session=62cd7ab1b05ec628ee1dd447462685a5; path=/; HttpOnly
Content-Length: 109
link_to_datafile
```

To start downloading the datafile via curl, you should perform the following operation:

```
curl -C - -O link_to_datafile
```

Check Job State

Purpose

This operation will provide you with the current job status and job outcome.

URL: link_to_server/jobs/job_id.xml

Request Type: GET

Request Parameters

None

Response Parameters

Parameters	Description
access-id	ID of the access used with this job
archive-output	Specifies whether the output datafiles should be packaged and compressed Type: boolean
argument-string	Specifies an argument string for the executable
description	Job description
executable-id	ID of the executable used with the job
id	Job ID
instance-type-id	ID of the instance type used with the job
job-outcome-id	Output datafile ID
license-id	ID of the license used with this job
name	Job name
nodes	Specifies the number of instances used to execute this job Type: integer
previous-job-id	ID of the previous job executed in a pipeline
reuse-inputs-from-job-id	Specifies the job ID to reuse the input datafiles from
start-immediately	Defines whether a job shall be started immediately or wait until there are free instances to reuse
status	Job execution status

stderr-file-name	Defines the standard error filename
stdout-file-name	Defines the standard output filename
stop-reason	Defines the reason for interruption of job execution
storage-region-id	Defines the ID of the region where the job datafiles are stored
use-ebs-ami	Specifies whether an elastic block store-backed Aamzon machine image should be used Type: boolean
used-core-runtime	Defines the used time length of job execution in conversion to one instance core in hours Type: integer
used-runtime	Defines the used time length of job execution in hours Type: integer
used-storage	Defines the storage space used for the job datafiles in bytes Type: integer
user-id	ID of the user submitting a job

Example Request

```
curl -k -u email:password link_to_server/jobs/a087e4d4-cf1b-4956-8c8a-6fab4d34c2e8.xml
```

Example Response

```
<job>
  <access-id>019f3fd3-6e15-4405-8ba4-9a9a3193f060</access-id>
  <archive-output type="boolean">false</archive-output>
  <argument-string></argument-string>
  <description></description>
  <executable-id>3530f805-8576-4a4a-add6-72a5d896e2d3</executable-id>
  <id>a087e4d4-cf1b-4956-8c8a-6fab4d34c2e8</id>
  <instance-type-id>711f4951-3c45-4ce8-86a2-a27cd653d912</instance-type-id>
  <job-outcome-id>23305791-c3d0-4281-99b0-60e25daa7033</job-outcome-id>
  <license-id>31bfb9fb-c65d-412f-9f22-2e7fe9922cd4</license-id>
  <name>Gromacs Job</name>
  <nodes type="integer">1</nodes>
  <previous-job-id></previous-job-id>
  <reuse-inputs-from-job-id nil="true"></reuse-inputs-from-job-id>
  <start-immediately type="boolean">true</start-immediately>
  <status>completed</status>
  <stderr-file-name>job.err</stderr-file-name>
  <stdout-file-name>job.out</stdout-file-name>
  <stop-reason nil="true"></stop-reason>
  <storage-region-id>2c4a5365-95cd-449f-b40d-36da48c93118</storage-region-id>
  <use-ebs-ami type="boolean">true</use-ebs-ami>
  <used-core-runtime type="integer">2</used-core-runtime>
  <used-runtime type="integer">1</used-runtime>
  <used-storage type="integer">0</used-storage>
  <user-id>fa87673a-b971-4ef2-b1a6-df38b13816b6</user-id>
</job>
```

Sitzungsprotokolle

Wöchentliche Fortschrittsbesprechungen

Datum: 28.09.2011

Ort: HSR, Rapperswil

Teilnehmer:

- Prof. Dr. Luc Bläser, Betreuer der Bachelorarbeit
- Patrice Müller, Projektteam
- Anita Hollenstein, Projektteam

Traktanden:

1. Präsentation und Besprechung des Projektablaufs
2. Festlegen eines Termins für wöchentliche Meetings
3. Gegenleser
4. Zusätzlicher Bildschirm für Bachelorarbeitszimmer
5. Zugang zum HSR-Cluster

Nr	Beschlüsse
1	Projekt ist bereits zu detailliert geplant. Als Erstes soll eine grobe Übersicht erstellt werden. Diese soll in Form von Meilensteinen Termine für wichtige Entscheidungen und Ergebnisse darstellen. Die Grobanalyse soll detailliert ausfallen und Architektur, Workflow, nichtfunktionale Anforderungen sowie die einzelnen Aspekte beschreiben. Ein weiterer wichtiger Punkt sind die Deliverables inkl. Risikoanalyse. Zum Beispiel sollen die Up- / Downloadgrössen überprüft werden. Weiter soll zum Beispiel in Erfahrung gebracht werden, was bei einem Netzunterbruch passiert. Muss sich der User neu einloggen?
2	Neu wird jede Woche auch am Dienstag um 13.00 Uhr ein Slot für Besprechungen freigehalten. Diese Besprechungen sind nicht zwingen, sollen jedoch bei Bedarf eine von der ASCOMP unabhängige Besprechung zwischen Betreuer und Studenten ermöglichen.
3	Der Gegenleser ist noch nicht bekannt und wird erst später festgelegt.
4	Ein zusätzlicher Bildschirm wird von Herr Bläser über Herr Rehmann organisiert.
5	Herr Bläser erkundigt sich bei Herrn Nordborg über Zugang zum HSR-Cluster.

Datum: 04.10.2011

Ort: HSR, Rapperswil

Teilnehmer:

- Prof. Dr. Luc Bläser, Betreuer der Bachelorarbeit
- Patrice Müller, Projektteam
- Anita Hollenstein, Projektteam

Traktanden:

1. Protokollbesprechung
2. Beurteilung Projektübersicht
3. Beurteilung Grobanalyse
4. Vorstellungen für Besprechung mit ASCOMP

Nr	Beschlüsse
1	-
2	Priorisierung ist zu früh → 2-stufige Grobanalyse machen
3	2-stufige Grobanalyse: - Zuerst Grobstudie mit Risiken und mehreren Lösungsmöglichkeiten - Danach Dossier mit weiteren Detailvorschlägen, aufzeigen was vorhanden ist, Priorisieren → Ende Oktober
4	Strukturierte Besprechung - Schwerpunkte aufzeigen - Fokus bestimmen: UI / Workflow-Support / Architekturdurchstich - Visualisierung lokal oder online - Login (Security) - Datenablage → detaillierte Frageliste erstellen

Datum: 25.10.2011

Ort: HSR, Rapperswil

Teilnehmer:

- Prof. Dr. Luc Bläser, Betreuer der Bachelorarbeit
- Patrice Müller, Projektteam
- Anita Hollenstein, Projektteam

Traktanden:

1. Check Meilenstein: Prototyp & Grobstudie
2. Nächste Schritte
3. Besuch ASCOMP vom 26.10.2011
4. Gegenleser

Nr	Beschlüsse
1	Aufgrund der Infrastukturprobleme, die aufgrund der Sicherheitspolicy im HSR-Netz entstanden sind, ist die Fertigstellung des Prototyps in Verzug. Weil sich der HPC der HSR hinter der DMZ des HSR-Netzes befindet, musste eine Lösung, die einen zusätzlichen IIS Server in der HSR-DMZ erfordert, erarbeitet werden. Fertiggestellt für Prototyp: - Poll-Service, der File von IIS auf HPC lädt (mit Installationsfile) - SL-Applikation mit Fileupload auf Server Ausstehend für Prototyp: - Poll-Service muss von einem HPC-Verantwortlichem auf dem HPC installiert werden (Installationsfile des Services steht bereit) - Verbindung zwischen SL-Applikation und IIS
2	Nächste Schritte: - Fertigstellung des Prototyps - Dokumentation der Erkenntnisse und Lösungen, da im Moment der Projektstand und die Projektstruktur nicht klar ersichtlich sind. - Treffen mit den Herren Nordborg und Baros vom HPC-Institut um Poll-Service und den Stand ihrer Portierung des TransAT-Solvers auf den HPC zu besprechen. Neue Beschlüsse: - Linux-Webserver von ASCOMP wird vorübergehend weggelassen. Das Hosting der Silverlight-Applikation auf Apache würde zu zusätzlichen Aufwänden führen, die für die Machbarkeitsstudie nicht von Bedeutung sind.
3	Besuch wird um eine Woche verschoben, da der Prototyp in Verzug ist.
4	Gegenleser ist Prof. Peter Sommerlad und der externe Experte Dr. Janos Zatonyi

Datum: 31.10.2011

Ort: HSR, Rapperswil

Teilnehmer:

- Prof. Dr. Luc Bläser, Betreuer der Bachelorarbeit
- Patrice Müller, Projektteam
- Anita Hollenstein, Projektteam

Traktanden:

1. Rückmeldung von Herr Bläser zu Projektfortschritt
2. Besprechung der Schwerpunkte

Nr	Beschlüsse
1	Besprochene Punkte: - Codereview für den ersten Prototyp - Dokumente: Die Dokumente werden fortlaufend erstellt, sind jedoch noch zu sehr Entwurf ähnlich um diese für einen Review abzugeben.
2	Für das Setzen von Schwerpunkten ist es noch zu früh. Als erstes soll als Basis ein Prototyp erstellt werden, der vorzeigt, was möglich ist und einzelne Parametereingaben zulässt.

Datum: 09.11.2011

Ort: HSR, Rapperswil

Teilnehmer:

- Prof. Dr. Luc Bläser, Betreuer der Bachelorarbeit
- Patrice Müller, Projektteam
- Anita Hollenstein, Projektteam

Traktanden:

1. Besprechung Dokumentation
2. Weiteres Vorgehen

Nr	Beschlüsse
1	- Weniger Details, mehr Konzeptschwerpunkte - Grafiken detailliert kommentieren - Schrittweiser Beschrieb des Simulationsworkflows fehlend Bestes Vorgehen: - Was muss gelöst werden? - Wie wurde es gelöst? - Wieso wurde es so gelöst?
2	- Fertigstellung des SAD - Prototyp mit Parametereingabe: o Öffnen eines Projekts o Eingabe von Parametern o Upload der Projektdateien o Solver-Mock o Download des Lösungsfiles Am Montag 14.11. wird der Projektplan anhand des Fortschritts ab heute 09.11. so angepasst, dass ein definitiver Termin für die Fertigstellung des SADs und des Prototyps festgelegt werden kann.

Datum: 16.11.2011

Ort: HSR, Rapperswil

Teilnehmer:

- Prof. Dr. Luc Bläser, Betreuer der Bachelorarbeit
- Patrice Müller, Projektteam
- Anita Hollenstein, Projektteam

Traktanden:

1. Zwischenstand Dokumentation
2. Zwischenstand Prototyp
3. Schwerpunkte

Nr	Beschlüsse
1	Dokumentation: - Use Cases und Requirements besser trennen in Funktionale Anforderungen, Nicht funktionale Anforderungen und erst dann Use Cases - Architektur- und Softwaredesign besser unterscheiden - Highlevel und Details in der Architektur besser unterscheiden - Visualisierung erwähnen - Risiken und Schwierigkeiten im Technischen Bericht auflisten - Threads erkären → Übersicht

Datum: 22.11.2011

Ort: HSR, Rapperswil

Teilnehmer:

- Prof. Dr. Luc Bläser, Betreuer der Bachelorarbeit
- Patrice Müller, Projektteam
- Anita Hollenstein, Projektteam

Traktanden:

1. Fortschrittsbesprechung Dokumentation
2. Festlegen von Vertiefungspunkten

Nr	Beschlüsse
1	Die folgenden Vertiefungspunkte wurden festgelegt, um diese in der morgigen Sitzung bei der ASCOMP zu priorisieren: 1. Backend: Anbindung an das Cloudbroker-Interface 2. UI Workflow: Verbesserte Usability 3. Parameter: Erweiterungskonzept für flexibel erweiterbare Parameter 4. Job Monitoring: Feedback von der Cloudschnittstelle um den Fortschritt einer laufenden Berechnung zu erhalten 5. Benutzerverwaltung: Multi-User Betrieb 6. Projektverwaltung: Persistente Projektverwaltung über Datenbank

Datum: 30.11.2011

Ort: HSR, Rapperswil

Teilnehmer:

- Prof. Dr. Luc Bläser, Betreuer der Bachelorarbeit
- Patrice Müller, Projektteam
- Anita Hollenstein, Projektteam

Traktanden:

1. Fortschrittsbesprechung

Nr	Beschlüsse
1	Implementierung: - Cloudbroker ist noch in der Testphase Dokumentation: - Prototyp soll genau beschrieben werden → was funktioniert, was nicht?

Datum: 07.12.2011

Ort: HSR, Rapperswil

Teilnehmer:

- Prof. Dr. Luc Bläser, Betreuer der Bachelorarbeit
- Patrice Müller, Projektteam
- Anita Hollenstein, Projektteam

Traktanden:

1. Codereview

Nr	Beschlüsse
1	Verschiedene Verbesserungsmöglichkeiten wurden im Code gefunden. Es wird ein Refactoring oder evtl. Codeanpassungen in den bemängelten Codeteilen durchgeführt.

Datum: 13.12.2011

Ort: HSR, Rapperswil

Teilnehmer:

- Prof. Dr. Luc Bläser, Betreuer der Bachelorarbeit
- Patrice Müller, Projektteam
- Anita Hollenstein, Projektteam

Traktanden:

1. Aufzeigen des Fortschritts
2. Codereview: Kontrolle der Überarbeitungen
3. Nächster Besuch bei der ASCOMP

Nr	Beschlüsse
1	Der nächste Besuch bei der ASCOMP findet voraussichtlich aus Zeitgründen erst im Neuen Jahr, also nach Abgabe der Bachelorarbeit statt.

Fortschrittsbesprechung bei der ASCOMP

Datum: 05.10.2011

Ort: ASCOMP, Zürich

Teilnehmer:

- Daniel Caviezel, ASCOMP
- **Narayanan Chidu, ASCOMP**
- Prof. Dr. Luc Bläser, Betreuer der Bachelorarbeit
- Patrice Müller, Projektteam
- Anita Hollenstein, Projektteam

Traktanden:

1. Besprechung Fragekatalog
2. Diverses
3. Weiteres Vorgehen

Nr	Beschlüsse
1	Sicherheit: - Die Sicherheit bei der Übertragung soll gewährleistet werden Linzenz Management: - Der Vorschlag eines Logins wurde begrüsst TransAT: - Diverse Fragen zu den Files wurden geklärt. (Wozu ist welches File, welche Files werden vom Solver benötigt, welche sind nur für das UI, …) o Vom Solver benötigte Files: *.f90, *.bc, *.grda, *.ls, properties.dat, *.inp - Die Grösse des *.grda Files kann bis GB gehen - Lösungsmöglichkeit für Vorgehensweise mit grossen Files: *Selektiver Download* Fokus: - Der Vorschlag eines Roundtrips für einen simplen Architekturdurchstich wurde angenommen Datenablage: - Daten sollen eher auf Webserver-Ebene als in der Cloud abgelegt werden Windows Portierung: - Die Silverlight-Applikation soll nicht die Windows-Portierung sein, sondern von mehreren Plattformen aus zugänglich sein. → **Risiko Plattformunterschiede**
2	Frage von ASCOMP: Wieso Silverlight?

	- Vorteil: Thin-Client - Nachteil: Linux-Benutzer evtl. nicht auf dem aktuellen Stand mit Moonlight
3	Weiteres Vorgehen - Erstellung von Workflow und Architektur - Webservice auf ASCOMP-Server (erst später)

Datum: 02.11.2011

Ort: ASCOMP, Zürich

Teilnehmer:

- Daniel Caviezel, ASCOMP
- **Narayanan Chidu, ASCOMP**
- **Dr. Wibke Sudholt, Cloudbroker**
- **Nicola Fantini, Cloudbroker**
- Prof. Dr. Luc Bläser, Betreuer der Bachelorarbeit
- Patrice Müller, Projektteam
- Anita Hollenstein, Projektteam

Traktanden:

1. Vorstellung Cloudbroker
2. Präsentation Prototyp
3. Weiteres Vorgehen

Nr	Beschlüsse
1	Cloudbroker erweist sich als sehr geeignet für unsere Arbeit: - Das Interface wird technologieunabhängig über RESTful Services angesprochen - Es besteht bereits eine Installation des TransAT Solvers auf der Cloud - Der bisherige Ansatz ein Zip-File an die Cloud zu senden ist auch hier möglich und sogar sinnvoll Weiteres Vorgehen: - Bevor wir uns für die Verwendung von Cloudbroker entscheiden, müssen wir intern die Projektplanung überprüfen und abklären, ob das Einbinden von Cloudbroker zeitlich möglich ist.
2	Das Userinterface sowie die benötigte Architektur für den Durchstich zum HSR-HPC-Cluster wurden der ASCOMP vorgestellt. - Das Userinterface mit dem Ansatz einer Projektverwaltung/-übersicht und der Unterstützung des bisherigen Workflows ist auf anklang gestossen. - Auch für die Netzwerkarchitektur wurde grosses Interesse gezeigt, da das aufgetretene Problem mit der Kommunikation ein allgemeines Problem der Cloud-/Cluster-Anbieter zu sein scheint.

3	Das weitere Vorgehen sind die folgenden Schritte: - Dokumentation der Architektur und des UI um diese der ASCOMP vorzulegen - Vertiefung und Dokumentation des Konzepts - Fortführung der UI-Entwicklung

Datum: 23.11.2011

Ort: ASCOMP, Zürich

Teilnehmer:

- **Dr. Jamel Lakehal, ASCOMP**
- Daniel Caviezel, ASCOMP
- **Narayanan Chidu, ASCOMP**
- Prof. Dr. Luc Bläser, Betreuer der Bachelorarbeit
- Patrice Müller, Projektteam
- Anita Hollenstein, Projektteam

Traktanden:

1. Vorstellung von Roundtrip und UI
2. Priorisierung der Vertiefungspunkte

Nr	Beschlüsse
1	Das ASCOMP Team war sehr erfreut an der Vorstellung des Zwischenstandes. Dieser entspricht ihren Vorstellungen.
2	Folgende Vertiefungspunkte wurden vorgeschlagen: 7. Backend: Anbindung an das Cloudbroker-Interface 8. UI Workflow: Verbesserte Usability 9. Parameter: Erweiterungskonzept für flexibel erweiterbare Parameter 10. Job Monitoring: Feedback von der Cloudschnittstelle um den Fortschritt einer laufenden Berechnung zu erhalten 11. Benutzerverwaltung: Multi-User Betrieb 12. Projektverwaltung: Persistente Projektverwaltung über Datenbank Die Priorisierung wurde auf die Punkte 1 + 2 gesetzt. Alle anderen Punkte sind in der weiteren Entwicklung zu vernachlässigen.

Zeiterfassung

Week	Date	User	Activity	Issue	Subject	Hours	Comment
1	19.09.2011	Anita Hollenstein	Infrastructure	99	Dokumentstruktur	1	""
1	19.09.2011	Anita Hollenstein	Infrastructure	104	Installation diverser Programme	6	""
1	19.09.2011	Patrice Müller	Infrastructure	104	Installation diverser Programme	6	""
1	20.09.2011	Anita Hollenstein	Analysis	111	Projektplanung	2	""
1	20.09.2011	Anita Hollenstein	Infrastructure	103	Git-Plugin in Visual Studio	6	""
1	20.09.2011	Patrice Müller	Infrastructure	101	Git-Repository eröffnen	12	Gab grosse Probleme mit dem Git-Repo und der Integrierung mit Redmine
1	20.09.2011	Patrice Müller	Infrastructure	100	Linux-Benutzer einrichten	2	Gab kleinere Probleme mit dem Skeleten. Einfach das Home-Dir selber erstellen
1	21.09.2011	Anita Hollenstein	Analysis	111	Projektplanung	3	""
1	21.09.2011	Anita Hollenstein	Infrastructure	107	Unklarheiten beseitigen	0.5	""
1	21.09.2011	Anita Hollenstein	Analysis	109	Diagramm erstellen	1	""
1	21.09.2011	Patrice Müller	Infrastructure	102	SSH-Konfiguration auf Server	8	Probleme mit Putty und dem Git-Bash auf Windows!
1	21.09.2011	Patrice Müller	Analysis	108	Workflow analysieren	1.5	""
1	23.09.2011	Patrice Müller	Infrastructure	110	Redmine Vorbereitung	2	""
1	23.09.2011	Patrice Müller	Infrastructure	98	Dropbox	0.5	""
1	23.09.2011	Patrice Müller	Analysis	111	Projektplanung	2	""
1	23.09.2011	Patrice Müller	Infrastructure	98	Dropbox	0.5	""
2	26.09.2011	Anita Hollenstein	Analysis	111	Projektplanung	4	""
2	27.09.2011	Anita Hollenstein	Analysis	111	Projektplanung	4	""
2	27.09.2011	Anita Hollenstein	Infrastructure	99	Dokumentstruktur	1	Festlegen von Layouts
2	27.09.2011	Anita Hollenstein	Documentation	104	Installation diverser Programme	2	Dokumentation der Infrastruktur
2	27.09.2011	Patrice Müller	Analysis	108	Workflow analysieren	3	""
2	27.09.2011	Patrice Müller	Analysis	111	Projektplanung	4	""
2	28.09.2011	Anita Hollenstein	Documentation	142	Allgemeine Dokumentationen	1	"Erstellen einer Vorlage für Protokolle, Protokollierung der heutigen Besprechung"
2	28.09.2011	Anita Hollenstein	Analysis	112	Themen identifizieren	2	""
2	28.09.2011	Anita Hollenstein	Documentation	139	Besprechungen Sprint 1	1	Fortschrittsbesprechung mit Betreuer
2	28.09.2011	Patrice Müller	Analysis	142	Allgemeine Dokumentationen	3	""
3	03.10.2011	Anita Hollenstein	Analysis	112	Themen identifizieren	2	""
3	03.10.2011	Anita Hollenstein	Analysis	141	Überarbeitung des Projektplans	2	""
3	03.10.2011	Anita Hollenstein	Analysis	113	Relevanz-Analyse	2	""
3	03.10.2011	Anita Hollenstein	Analysis	105	Risikoanalyse	2	""
3	03.10.2011	Patrice Müller	Analysis	117	TransAt Installieren	5	Habe auch schon Tutorials gemacht
3	03.10.2011	Patrice Müller	Analysis	119	Definition der Problem Domain	4	""
3	04.10.2011	Anita Hollenstein	Analysis	119	Definition der Problem Domain	1	Überarbeitung der Architekturübersicht
3	04.10.2011	Anita Hollenstein	Documentation	139	Besprechungen Sprint 1	2	"Besprechung Vorgehen bei Ascomp-Besprechung vom 5.10. Vorbereitung der Ascomp-Sitzung "
3	04.10.2011	Anita Hollenstein	Analysis	118	TransAt Manuals Lesen	4	Lesen der Manuals und Durchführung eines Tutorials
3	04.10.2011	Patrice Müller	Development	186	Fileupload HPC-->IIS	5	""
3	05.10.2011	Anita Hollenstein	Analysis	139	Besprechungen Sprint 1	1.5	Vorbereitung der Besprechung mit Ascomp
3	05.10.2011	Anita Hollenstein	Documentation	142	Allgemeine Dokumentationen	0.5	""
3	05.10.2011	Anita Hollenstein	Documentation	139	Besprechungen Sprint 1	1.5	Besprechung bei der Ascomp
3	05.10.2011	Anita Hollenstein	Analysis	119	Definition der Problem Domain	2	Überarbeitung der Architekturübersicht
3	05.10.2011	Patrice Müller	Analysis	118	TransAt Manuals Lesen	4	Ist für diesen Sprint mal fertig!
3	07.10.2011	Anita Hollenstein	Documentation	139	Besprechungen Sprint 1	1	Reinschreiben des Protokolls vom 5.
3	07.10.2011	Anita Hollenstein	Documentation	105	Risikoanalyse	4	Grobanalyse mit Risiken und Lösungsmöglichkeiten
4	10.10.2011	Anita Hollenstein	Analysis	143	Einarbeiten in den HSR HPC	1	Besprechung des HPC mit V. Baros
4	10.10.2011	Anita Hollenstein	Analysis	143	Einarbeiten in den HSR HPC	4	Durcharbeiten der Links und desTutorials von V. Baros
4	10.10.2011	Anita Hollenstein	Documentation	168	Grobanalyse	2	Dokumentation der Grobanalyse

4	10.10.2011	Patrice Müller	Infrastructure	185	HSR-Netzwerk durchstich	12	""
4	10.10.2011	Patrice Müller	Development	143	Einarbeiten in den HSR HPC	4	""
4	10.10.2011	Patrice Müller	Documentation	139	Besprechungen Sprint 1	1.5	""
4	10.10.2011	Patrice Müller	Analysis	158	Zugriff von einem nicht HSR computer	1.5	Sitzung mit Vladimir
4	10.10.2011	Patrice Müller	Analysis	158	Zugriff von einem nicht HSR computer	1	""
4	11.10.2011	Anita Hollenstein	Development	122	File hochladen	2	Beginn der Implementation eines Fileuploads
4	11.10.2011	Anita Hollenstein	Analysis	169	Einarbeitung IIS Server	2.5	Informationssammlung zum IIS
4	11.10.2011	Anita Hollenstein	Documentation	168	Grobanalyse	2	Fortsetzung Dokumentation Grobanalyse
4	11.10.2011	Patrice Müller	Development	166	Einarbeitung Silverlight mit WCF	8.5	WCF Eingelesen und wieder aufgefrischt
4	11.10.2011	Patrice Müller	Analysis	122	File hochladen	2	Workflow diagramm zur entscheidung welche Files hochgeladen werden sollen
4	12.10.2011	Anita Hollenstein	Development	122	File hochladen	2	Fortsetzung Fileupload
4	12.10.2011	Anita Hollenstein	Analysis	166	Einarbeitung Silverlight mit WCF	4	Einlesen in WCF
4	12.10.2011	Anita Hollenstein	Documentation	167	Besprechungen Sprint 2	1	Besprechung mit Herr Bläser
4	12.10.2011	Patrice Müller	Documentation	167	Besprechungen Sprint 2	1	""
4	13.10.2011	Anita Hollenstein	Analysis	166	Einarbeitung Silverlight mit WCF	2	Informationssuche zum SL-WCF Datenupload
5	17.10.2011	Anita Hollenstein	Development	122	File hochladen	8	Implementation eines einfachen UIs mit Fileupload auf Server
5	17.10.2011	Patrice Müller	Infrastructure	185	HSR-Netzwerk durchstich	5	""
5	17.10.2011	Patrice Müller	Infrastructure	183	IIS in der DMZ	8.4	"Das IIS GUI wurder mir zum Verhängnis, habe einen ganzen Tag daran verloren => Fand die IIS-Module nicht"
5	18.10.2011	Anita Hollenstein	Development	178	UI	3	Aufbau eines MVVM Grundgerüsts mit dem MVVM Light Toolkit
5	18.10.2011	Anita Hollenstein	Development	122	File hochladen	5	"Verbesserung des Fileuploads (mehrere Files gleichzeitig, grosse Files)"
5	18.10.2011	Patrice Müller	Development	184	Filewatcher	4	Das HSR-Netz war einem Mal wieder im weg => Netshare in die DMZ vom HSR-Secure war nicht offen habe wieder Zeit verloren um dies rauszufinden
5	19.10.2011	Anita Hollenstein	Development	178	UI	7	Erweiterung des UIs und den zugehörigen Funtkionen
5	19.10.2011	Anita Hollenstein	Documentation	167	Besprechungen Sprint 2	0.5	Kurze Fortschrittsbesprechung
5	19.10.2011	Patrice Müller	Documentation	167	Besprechungen Sprint 2	0.5	""
5	20.10.2011	Anita Hollenstein	Development	122	File hochladen	3	Recherche betreffend Zip erstellen und benötigtem IsolatedStorage
6	24.10.2011	Anita Hollenstein	Development	179	Erweiterung des Fileuploads	2	Erweiterung für Upload von grossen Files
6	24.10.2011	Anita Hollenstein	Development	179	Erweiterung des Fileuploads	4	"Zusammen der Datein zu Zip Upload des Zips"
6	24.10.2011	Anita Hollenstein	Development	180	Threading	2	Auslagern von CreateZip und Read Zip in separate Threads
6	24.10.2011	Patrice Müller	Development	184	Filewatcher	8.4	Es war nicht ganz einfach sich über das Netzwerk mit einem anderen Benutzer zu Indentifizieren (P/Invoke net use add)
6	25.10.2011	Anita Hollenstein	Documentation	181	Besprechungen Sprint 3	1	Besprechung zum Fortschritt
6	25.10.2011	Anita Hollenstein	Development	180	Threading	2	Auslagern von Fileupload in separaten Workerthread
6	25.10.2011	Anita Hollenstein	Development	179	Erweiterung des Fileuploads	4	Veranschaulichung des Fortschritts
6	25.10.2011	Anita Hollenstein	Development	182	MVVM	1	Austausch von Messages um enge Bindung zwischen Views und ViewModels zu vermeiden
6	25.10.2011	Patrice Müller	Analysis	169	Einarbeitung IIS Server	2.5	""
6	26.10.2011	Anita Hollenstein	Documentation	193	Projektplanung	2	Überarbeitung der Projektplanung
6	26.10.2011	Anita Hollenstein	Development	180	Threading	1.5	Problemsuche: Worker endet zu früh
6	26.10.2011	Anita Hollenstein	Development	180	Threading	2	Threading des Uploads optimiert
6	26.10.2011	Anita Hollenstein	Development	182	MVVM	2	Messages von Upload Klasse zu UploadViewModel
6	26.10.2011	Patrice Müller	Development	184	Filewatcher	6	""
6	26.10.2011	Patrice Müller	Development	193	Projektplanung	2	Projektplan wurde aktuallsiert
6	27.10.2011	Anita Hollenstein	Development	196	Publish to IIS	2	"Publish auf IIS Fehlersuche: Files werden nicht abgelegt"
6	27.10.2011	Anita Hollenstein	Development	179	Erweiterung des Fileuploads	0.5	Fehlerbehebung vom FileUpload Progress
6	27.10.2011	Anita Hollenstein	Documentation	181	Besprechungen Sprint 3	0.5	Reinschreiben des Protokolls
6	27.10.2011	Anita Hollenstein	Development	196	Publish to IIS	2	Problembehebung von IsolatedStorage
6	28.10.2011	Anita Hollenstein	Development	196	Publish to IIS	1	"Fertigstellung, Problem waren unterschiedliche ASP.NET Versionen von Projekt und IIS"

6	28.10.2011	Anita Hollenstein	Development	194	Paperprototype	1.5	Paperprototype basierend auf TransATUI mit erweiterter Projektverwaltung
6	30.10.2011	Anita Hollenstein	Development	197	UI	1	ProjectView ohne logische Funktionen
7	31.10.2011	Anita Hollenstein	Development	197	UI	2	Implementation Projektübersicht
7	31.10.2011	Anita Hollenstein	Documentation	181	Besprechungen Sprint 3	1	Fortschrittsbesprechung
7	31.10.2011	Anita Hollenstein	Documentation	181	Besprechungen Sprint 3	2	"Vorbereitung Projektfortschrittsbesprechung, Zusammenstellung der Schwerpunkte"
7	01.11.2011	Anita Hollenstein	Development	197	UI	6	Implementation UI basierend auf Paperprototype
7	01.11.2011	Anita Hollenstein	Development	197	UI	2	Erstellen eines UI Designs mit Logo
7	01.11.2011	Patrice Müller	Analysis	181	Besprechungen Sprint 3	1.5	""
7	01.11.2011	Patrice Müller	Development	186	Fileupload HPC-->IIS	5	Noch nicht besser
7	01.11.2011	Patrice Müller	Documentation	195	Fortführung der Grobanalyse	5	Netzwerk Grafik
7	02.11.2011	Anita Hollenstein	Documentation	181	Besprechungen Sprint 3	2	Acomp Besprechung: Cloudbroker und Vorstellung Prototyp
7	02.11.2011	Anita Hollenstein	Documentation	181	Besprechungen Sprint 3	1	Vorbereitung der Ascompbesprechung
7	02.11.2011	Anita Hollenstein	Development	197	UI	3	Erweiterung des UIs
7	02.11.2011	Patrice Müller	Development	207	Vorbereitung Sitzugn Ascomp & Cloudbroker	4	"war leider für nichts, konnte alles wieder verwerfen"
7	02.11.2011	Patrice Müller	Analysis	181	Besprechungen Sprint 3	2.5	""
7	03.11.2011	Patrice Müller	Development	205	FileMover	9	""
7	06.11.2011	Anita Hollenstein	Development	197	UI	2	Verbesserung der bisherigen GUI Features
8	07.11.2011	Anita Hollenstein	Development	197	UI	8	"Weiterentwicklung des GUIs, Mock-Projektdaten"
8	07.11.2011	Patrice Müller	Documentation	188	Webservice-Konzept	7	""
8	08.11.2011	Anita Hollenstein	Development	191	Webservice	1.5	Duplex Service zur ermöglichung einer Pull-Notification von Server zu Client
8	08.11.2011	Anita Hollenstein	Development	197	UI	1.5	Allgemeine Verbesserungen im GUI Code
8	08.11.2011	Anita Hollenstein	Development	197	UI	5	"Weiterentwicklung, neue View für Parametereingabe"
8	08.11.2011	Patrice Müller	Documentation	188	Webservice-Konzept	8	""
8	09.11.2011	Anita Hollenstein	Documentation	220	Besprechungen Sprint 4	1	Fortschrittsbesprechung
8	09.11.2011	Anita Hollenstein	Development	190	Filewatcher	1	Erweiterung des Duplexservices mit Filewatcher
8	09.11.2011	Anita Hollenstein	Development	179	Erweiterung des Fileuploads	1.5	Codeverbesserungen im Fileup-/Download
8	09.11.2011	Anita Hollenstein	Development	179	Erweiterung des Fileuploads	1.5	Verbesserung des Filedownloads
8	09.11.2011	Anita Hollenstein	Development	191	Webservice	3	Weiterentwicklung des Duplexservice
8	10.11.2011	Patrice Müller	Documentation	221	Software Dokumentation	6	Allg. Arichtektur Implementation
8	11.11.2011	Anita Hollenstein	Development	191	Webservice	2	Fehlerbehebung im Duplex Service
8	12.11.2011	Anita Hollenstein	Documentation	221	Software Dokumentation	2	Dokumentation des Architekturaufbaus
8	13.11.2011	Anita Hollenstein	Documentation	221	Software Dokumentation	3	Fortführung Dokumentation Architekturaufbau
9	14.11.2011	Anita Hollenstein	Documentation	221	Software Dokumentation	8	"Dokumentation der Analyse, Design und Architektur"
9	14.11.2011	Patrice Müller	Development	222	FileParser	8.5	""
9	15.11.2011	Anita Hollenstein	Documentation	221	Software Dokumentation	8	Dokumentation Architektur und Implementation
9	15.11.2011	Patrice Müller	Development	223	FileWrite	8.5	""
9	16.11.2011	Anita Hollenstein	Documentation	220	Besprechungen Sprint 4	1.5	"Fortschrittsbesprechung Doku, Prototyp 2"
9	16.11.2011	Anita Hollenstein	Development	218	Weiterentwicklung des UI	5	"Problemsuche bei Duplex-Service, Lösungssuche für Solutionfiledownload"
9	16.11.2011	Anita Hollenstein	Development	218	Weiterentwicklung des UI	2	UI Design
9	16.11.2011	Patrice Müller	Testing	224	Mokup Bugfixing	8.5	""
9	19.11.2011	Anita Hollenstein	Development	218	Weiterentwicklung des UI	3	"Pojektmanagement, Upload einzelner lokaler Projektdateien"
9	20.11.2011	Anita Hollenstein	Development	218	Weiterentwicklung des UI	8	"Client-Pollservice, Projektmanagement"
10	21.11.2011	Anita Hollenstein	Development	218	Weiterentwicklung des UI	1	Korrektur im ProjectViewModel
10	21.11.2011	Anita Hollenstein	Documentation	221	Software Dokumentation	9	Überarbeitung ges. Technischer Bericht und Software Dokumentation Kapitel Implementation
10	21.11.2011	Patrice Müller	Testing	224	Mokup Bugfixing	6.5	Auf dem Computer PIN1206010 lief der Mock-up nicht obschon er getetst wurde. Bug gefunden und gefixt
10	21.11.2011	Patrice Müller	Documentation	221	Software Dokumentation	2.5	"Erste Mockup, Parser Dokumentation"
10	22.11.2011	Anita Hollenstein	Documentation	220	Besprechungen Sprint 4	1.5	Fortschrittsbesprechung
10	22.11.2011	Anita Hollenstein	Development	218	Weiterentwicklung des UI	3.5	Fehlersuche Isolated Storage
10	22.11.2011	Anita Hollenstein	Documentation	221	Software Dokumentation	3	Ausarbeitung Dokumentation

10	22.11.2011	Patrice Müller	Development	239	Connect to Cloudbroker	4	""
10	22.11.2011	Patrice Müller	Development	240	XML2Object - Object2XML	4	""
10	23.11.2011	Anita Hollenstein	Documentation	220	Besprechungen Sprint 4	1.5	Besprechung Ascomp
10	23.11.2011	Anita Hollenstein	Development	218	Weiterentwicklung des UI	3.5	Problembehebung Isolated Storage
10	23.11.2011	Patrice Müller	Infrastructure	239	Connect to Cloudbroker	8	""
10	26.11.2011	Anita Hollenstein	Development	218	Weiterentwicklung des UI	1.5	Projekt- und Storageverwaltung
10	27.11.2011	Anita Hollenstein	Development	218	Weiterentwicklung des UI	6	"Projektverwaltung mit DB-Mockup, Isolated Storage Verwaltung + Fehlersuche"
11	28.11.2011	Anita Hollenstein	Development	238	Allgemeine UI-Arbeiten	8	"Project Managent: Upload, Download und Löschen von ganzen Projekten"
11	28.11.2011	Patrice Müller	Development	239	Connect to Cloudbroker	8	Curl ausprobiert
11	29.11.2011	Anita Hollenstein	Design	237	Unterstützung des Workflows	4	Design Breadcrumb Objekte
11	29.11.2011	Anita Hollenstein	Development	238	Allgemeine UI-Arbeiten	4	Abschluss Projektmanagement
11	29.11.2011	Patrice Müller	Development	240	XML2Object - Object2XML	2	XDS ist zwar nett wenn es schnell gehen muss. Aber der Code ist nicht gerade schön geschweige den Leserlich
11	29.11.2011	Patrice Müller	Development	239	Connect to Cloudbroker	9	"X.509 Problematik, da Cloudbroker kein ""Trusted"" Zertfikat hat"
11	30.11.2011	Anita Hollenstein	Documentation	233	Besprechungen Sprint 5	0.5	Fortschrittsbesprechung
11	30.11.2011	Anita Hollenstein	Development	237	Unterstützung des Workflows	7	Breadcrumbs für Project-Workflow
11	30.11.2011	Patrice Müller	Documentation	235	Allgemeine Dokumentation	6	Codecleaning
11	30.11.2011	Patrice Müller	Development	239	Connect to Cloudbroker	8	""
11	01.12.2011	Anita Hollenstein	Development	237	Unterstützung des Workflows	2	Fortsetzung Breadcrumbsimplementation
11	02.12.2011	Anita Hollenstein	Development	238	Allgemeine UI-Arbeiten	5	UI Parametererweiterung
11	02.12.2011	Anita Hollenstein	Development	238	Allgemeine UI-Arbeiten	2.5	UI Erweiterung
11	02.12.2011	Patrice Müller	Development	244	Job-Erstellen	5	Jetzt muss man aus dem Objet noch den job erstellen
11	02.12.2011	Patrice Müller	Development	239	Connect to Cloudbroker	6	""
11	03.12.2011	Anita Hollenstein	Development	238	Allgemeine UI-Arbeiten	3	UI Verbesserung und Erweiterung
11	04.12.2011	Anita Hollenstein	Development	238	Allgemeine UI-Arbeiten	5	"UI Erweiterung, Darstellen der ersten Parameterwerte vom input-File im UI"
12	05.12.2011	Anita Hollenstein	Development	238	Allgemeine UI-Arbeiten	8	Parametereingaben im UI --> Schreiben in File
12	05.12.2011	Patrice Müller	Development	241	Job Hochladen	4	Probleme mit dem TransAt Solver Job Creation
12	05.12.2011	Patrice Müller	Development	241	Job Hochladen	4	Problem mit dem Post header
12	05.12.2011	Patrice Müller	Development	240	XML2Object - Object2XML	2	""
12	05.12.2011	Patrice Müller	Testing	245	Paser kann mit Kommentare umgehen	3	""
12	06.12.2011	Anita Hollenstein	Development	238	Allgemeine UI-Arbeiten	10	"Parameter mapping, File editing, Erweiterung des UI"
12	06.12.2011	Patrice Müller	Development	222	FileParser	3.5	BugFixing
12	06.12.2011	Patrice Müller	Development	240	XML2Object - Object2XML	6	""
12	07.12.2011	Anita Hollenstein	Development	236	Fortschrittsanzeigen	4	Fortschrittsanzeigen für Projektupload
12	07.12.2011	Anita Hollenstein	Development	233	Besprechungen Sprint 5	1.75	Besprechung des Fortschritts und Codereview
12	07.12.2011	Anita Hollenstein	Development	238	Allgemeine UI-Arbeiten	4.5	Schreiben der UI Parameter in das input File
12	07.12.2011	Patrice Müller	Development	239	Connect to Cloudbroker	10	""
12	08.12.2011	Anita Hollenstein	Development	238	Allgemeine UI-Arbeiten	3	Schreiben der UI Parameter in das input File
12	08.12.2011	Patrice Müller	Development	239	Connect to Cloudbroker	4	Still prolbems mit dem Header
12	08.12.2011	Patrice Müller	Development	239	Connect to Cloudbroker	9.5	""
12	09.12.2011	Anita Hollenstein	Development	238	Allgemeine UI-Arbeiten	8	"save and edit restrictions, MessageWindow, Copy Project"
12	09.12.2011	Patrice Müller	Development	239	Connect to Cloudbroker	8	""
12	09.12.2011	Patrice Müller	Development	241	Job Hochladen	4	""
12	10.12.2011	Anita Hollenstein	Development	238	Allgemeine UI-Arbeiten	5	"diverse UI Korrekturen, upload tar.gz"
12	11.12.2011	Anita Hollenstein	Development	238	Allgemeine UI-Arbeiten	5	"Diverse UI Korrekturen, solution download, save state"
12	11.12.2011	Patrice Müller	Development	239	Connect to Cloudbroker	12	""
13	12.12.2011	Anita Hollenstein	Development	246	Silverlight Client Refactoring	4	Codebereinigung
13	12.12.2011	Anita Hollenstein	Development	247	Silverlight Client Bugfixing	4	Behebung von Bugs im UI und Isolated Storage
13	12.12.2011	Patrice Müller	Development	239	Connect to Cloudbroker	8	Webservice zu Cloudbroker geschrieben
13	12.12.2011	Patrice Müller	Documentation	235	Allgemeine Dokumentation	1	""
13	12.12.2011	Patrice Müller	Development	242	Status von Job auslesen	1	Gibt einfache den ganzen Job zurück
13	13.12.2011	Anita Hollenstein	Development	246	Silverlight Client Refactoring	3	Refactoring des Clients
13	13.12.2011	Anita Hollenstein	Documentation	248	Poster	1	Vorbereitung des Posters

13	13.12.2011	Anita Hollenstein	Documentation	249	Abstract	2	Vorbereitung des Abstracts
13	13.12.2011	Anita Hollenstein	Documentation	250	Besprechungen Sprint 6	2	Besprechung zum Fortschritt und Code
13	13.12.2011	Patrice Müller	Documentation	248	Poster	1	""
13	13.12.2011	Patrice Müller	Documentation	249	Abstract	1	""
13	13.12.2011	Patrice Müller	Documentation	250	Besprechungen Sprint 6	2	""
13	14.12.2011	Anita Hollenstein	Documentation	249	Abstract	3	"Arbeit am Abstract, Erstellen von Grafiken"
13	14.12.2011	Anita Hollenstein	Development	253	Cloudbrokerservice mit SilverlightService verbinden	2	Verbinden von Cloudbroker mit SL Client
13	14.12.2011	Anita Hollenstein	Development	247	Silverlight Client Bugfixing	4	Beheben von Fehlern im Client
13	14.12.2011	Patrice Müller	Development	253	Cloudbrokerservice mit SilverlightService verbinden	8	""
13	15.12.2011	Anita Hollenstein	Documentation	249	Abstract	5	Fertigstellung des Abstracts
13	15.12.2011	Patrice Müller	Testing	252	CloudbrokerAPI	4	""
13	15.12.2011	Patrice Müller	Testing	251	CloudbrokerService	5	""
13	16.12.2011	Anita Hollenstein	Documentation	254	Allgemeine Dokumentation	1	Arbeit an der allgemeinen Dokumentation
13	16.12.2011	Anita Hollenstein	Development	253	Cloudbrokerservice mit SilverlightService verbinden	3	Verbinden von Cloudbroker mit SL Client
13	16.12.2011	Anita Hollenstein	Documentation	248	Poster	4	Arbeit am Poster
14	19.12.2011	Anita Hollenstein	Development	253	Cloudbrokerservice mit SilverlightService verbinden	3	Bugfixing
14	19.12.2011	Anita Hollenstein	Documentation	265	Kapitel Anforderungspezifkation	3	Überarbeitung des Kapitels
14	19.12.2011	Anita Hollenstein	Documentation	266	Analyse	2.5	Überarbeitung des Kapitels
14	19.12.2011	Patrice Müller	Documentation	256	Stand der Technik: Beschreibung Cloudbroker	2	""
14	19.12.2011	Patrice Müller	Documentation	256	Stand der Technik: Beschreibung Cloudbroker	2	
14	19.12.2011	Patrice Müller	Development	251	CloudbrokerService	4	
14	20.12.2011	Anita Hollenstein	Documentation	259	Kapitel Design abschliessen	6	Überarbeitung SL Client Doku
14	20.12.2011	Anita Hollenstein	Documentation	263	Überarbeitung Projektmanagement und Projektmonitoring	3	Überarbeitung Kapitel Projektmanagement
14	20.12.2011	Anita Hollenstein	Documentation	271	Anhang	2	"Anhang: Einfügen Sitzungsprotokolle, anpassung Dokumentstruktur"
14	20.12.2011	Patrice Müller	Documentation	260	Kapitel Implementation abschliessen	5	Überarbeitung des bestehnden
14	20.12.2011	Patrice Müller	Documentation	260	Kapitel Implementation abschliessen	2	
14	21.12.2011	Anita Hollenstein	Documentation	271	Anhang	1	Einfügen der PDF Anhänge
14	21.12.2011	Anita Hollenstein	Documentation	273	Korrektur	2	Korrekturlesen
14	21.12.2011	Anita Hollenstein	Documentation	274	Korrektur	0.5	Korrekturlesen Kapitel 1+2
14	21.12.2011	Anita Hollenstein	Documentation	258	Resultate	1	Dokumentation Erreichte Ziele
14	21.12.2011	Anita Hollenstein	Documentation	257	Evaluation	2	Dokumentation Risiken und Schwierigkeiten
14	21.12.2011	Anita Hollenstein	Documentation	263	Überarbeitung Projektmanagement und Projektmonitoring	2	Projektmonitoring
14	21.12.2011	Anita Hollenstein	Documentation	257	Evaluation	3	Arbeit am Kapitel Evaluation
14	21.12.2011	Patrice Müller	Documentation	259	Kapitel Design abschliessen	4	
14	21.12.2011	Patrice Müller	Documentation	260	Kapitel Implementation abschliessen	4	
14	22.12.2011	Anita Hollenstein	Documentation	263	Überarbeitung Projektmanagement und Projektmonitoring	2	Überarbeitung
14	22.12.2011	Anita Hollenstein	Documentation	275	Management Summary	2	Erstellen des Management Summary
14	22.12.2011	Anita Hollenstein	Documentation	258	Resultate	2	Persönlicher Bericht und Lessons Learned
14	22.12.2011	Anita Hollenstein	Documentation	260	Kapitel Implementation abschliessen	4	UI Design und Thread Design
14	22.12.2011	Patrice Müller	Documentation	262	Resultate und Weiterentwicklung	4	
14	22.12.2011	Patrice Müller	Documentation	261	Dokumentation des Testings	4	
14	23.12.2011	Anita Hollenstein	Documentation	274	Korrektur	1	Korrekturlesen
14	23.12.2011	Anita Hollenstein	Documentation	276	Abgabe	3	Letzte Kontrolle vom Dokument, Poster ausdrucken, CD brennen
14	23.12.2011	Patrice Müller	Documentation	268	Glossar	2	
14	23.12.2011	Patrice Müller	Documentation	267	Literaturverzeichnis	2	