



**Berner Fachhochschule**

Technik und Informatik

Software-Schule Schweiz

Diplomarbeit MAS-06-01.06

# SoapCalc

## Dokumentation

Klasse MAS-IT-06-01, 2008

### Abstract:

Im Rahmen der Mobilfunknetz- Planung und Optimierung fallen häufig rechenintensive Datenverarbeitungen an wie z.B. die Berechnung der Abdeckung der Schweizer Bevölkerung mit dem Mobilfunkdienst HSPA oder die Verarbeitung von georeferenzierten Daten (Messungen, GIS Layer usw.). Die Idee von SoapCalc ist es, diese Berechnungen auf verschiedene Server zu verteilen um die Verarbeitungszeit zu verkürzen und die Abläufe zu automatisieren. Das System ist aber frei konfigurierbar für beliebige rechenintensive Datenverarbeitungen.

For the radio network planning and optimization, computationally intensive data processings are often required, e.g. the calculation of the coverage of the Swiss population with the service HSPA or the processing of georeferencing data (Measures, GIS layer etc.). The goal of SoapCalc is to distribute these calculations on different servers in order to reduce the calculation time and to automate the workflow. However, the system is freely configurable for any computationally intensive calculations.

### Schlüsselwörter:

Verteiltes Rechnen, SOA, WCF

### Student:

Michael Berger, Mönchstrasse 33, 3600 Thun  
michael.berger@embe.ch, 079 506 07 69

### Betreuer:

Stefan Bigler, Enkom Inventis AG, Worbstrasse 225, 3073 Gümligen  
stefan.bigler@enkom.com, 031 950 42 31

### Experte:

Rolf Wenger, INFOBRAIN AG, Obere Zollgasse 75, 3072 Ostermundigen  
rolf.wenger@infobrain.com, 031 544 22 22

# Management Summary

Aufwendige Berechnungsaufgaben werden durch die Software SoapCalc auf verschiedene Server verteilt um die Verarbeitungszeit zu verkürzen und die Abläufe zu automatisieren. Die Aufgabe der Software SoapCalc ist es, diese Aufgaben zu verwalten, die Verteilung zu kontrollieren und die Aufträge auszuführen. Dem Benutzer steht eine Weboberfläche zur Verfügung, um das System zu bedienen.

Realisiert wurde das System mit dem .NET Framework. Dabei wurden die neuen Features von .NET 3.0 bzw. 3.5 eingesetzt. Konkret sind dies die WCF (.NET 3.0), LINQ (.NET 3.5) sowie die Neuerungen von ASP.NET. Anhand von Prototypen wurde zuerst geprüft, ob sich diese Technologien für diese Anwendung eignen. Damit wurden auch Erfahrungen für zukünftige Projekte gesammelt.

Als Designunterstützung wurden zwei Software Factories aus dem design & practice Umfeld von Microsoft eingesetzt. Die Web Service Software Factory erleichtert das Modellieren von WCF Diensten auf hohem Abstraktionsniveau. Die Web Client Software Factory unterstützt bei der Entwicklung von Webseiten. Beide setzen gängige Patterns ein, was in einer sauberen Architektur resultiert. Der Einsatz von Open Source Bibliotheken, beispielsweise einer Zip Bibliothek, einer Graphen Bibliotheken zur Prüfung von zirkulären Abhängigkeiten oder zusätzlichen Web Controls konnte die Funktionalität und die Benutzerführung ohne grossen Mehraufwand deutlich verbessern.

Auf der anderen Seite war aber gerade die Entwicklung der Weboberfläche sehr aufwändig. Der Einsatz von AJAX Komponenten steigert den Benutzerkomfort, ist aber mit einem erhöhten Entwicklungsaufwand verbunden. Die Erwartungen an den angekündigten Nachfolger Silverlight sind entsprechend hoch.

Die aktuelle Version eins läuft stabil und verfügt über alle nötigen Funktionalitäten zum produktiven Einsatz. Für alle Komponenten gibt es aber Verbesserungsmöglichkeiten, die in einer allfälligen Version zwei realisiert werden könnten.

# Inhaltsverzeichnis

1	Einleitung.....	1
1.1	Sinn und Zweck.....	1
1.2	Umfang und Anwendungsbereich.....	1
1.3	Definitionen und Abkürzungen.....	1
1.4	Glossar.....	2
1.5	Design- und Programmierkonvention.....	2
1.6	Abgrenzungen.....	2
1.7	Zeitplan.....	3
1.7.1	Milestones.....	3
2	Software Analyse.....	4
2.1	Grobübersicht mit MindMap.....	4
2.2	Schema.....	5
2.2.1	Auftragsserver.....	5
2.2.2	Berechnungsserver.....	5
2.2.3	Log Server.....	5
2.2.4	User Interface / Weboberfläche.....	5
2.3	Umfeld.....	6
2.3.1	Benutzer-Schnittstelle.....	6
2.3.2	Kommunikations-Schnittstelle.....	6
2.3.3	Software-Schnittstelle.....	6
2.3.4	Hardware.....	6
2.4	Änderungen gegenüber dem Pflichtenheft.....	7
2.4.1	Anpassungen von Anforderungen.....	7
2.4.2	Nicht erfüllte Anforderungen.....	7
2.5	Spezifische Anforderungen.....	8
2.5.1	Funktionen.....	8
2.5.1.1	Auftragsserver.....	8
2.5.1.2	Berechnungsserver.....	8
2.5.1.3	Log Server.....	9
2.5.1.4	User Interface.....	9
2.5.2	Mengengerüst.....	9
2.5.3	Business Objects.....	10
2.5.3.1	Auftragsdefinitionen.....	10
2.5.3.2	Aufträge.....	10
2.5.3.3	Auftragsvorlage.....	11
2.5.3.4	Sammelauftrag.....	11
2.5.3.5	Log Eintrag.....	11
2.5.3.6	Auslastungsreport.....	12
2.5.4	Datenbasis.....	12
2.5.4.1	Datenbank.....	12
2.5.4.2	File Store.....	13
2.5.5	Sicherheit.....	14
2.5.5.1	Komponenten.....	14
2.5.5.2	Benutzer.....	14
2.5.6	Kommunikation.....	14
2.6	Anwendungsfalldiagramme.....	15
2.6.1	Auftragsserver.....	15
2.6.2	Berechnungsserver.....	15
2.6.3	Log Server.....	16
2.6.4	Log Analyzer.....	16
2.6.5	Weboberfläche.....	16
2.6.5.1	Aufträge verwalten.....	16
2.6.5.2	Berechnungsserver.....	17
2.6.5.3	Administrationsbereich.....	17
2.7	Anwendungsfallbeschreibungen.....	18

---

2.8	Aktivitätsdiagramme .....	27
2.8.1	Auftragsserver.....	27
2.8.1.1	Neuer Job.....	27
2.8.1.2	Auftrag vorbereiten.....	28
2.8.1.3	Auftrag verteilen .....	29
2.8.1.4	Auswahl Berechnungsserver .....	30
2.8.2	Berechnungsserver.....	31
2.8.2.1	Abmeldung Berechnungsserver vom Auftragsserver .....	31
2.8.3	Log Server.....	32
2.8.3.1	Neuer Log Eintrag .....	32
2.8.3.2	Eintrag in DB speichern .....	32
3	Software Design .....	33
3.1	Einleitung in die Software Factories.....	33
3.1.1	Web Client Software Factory .....	33
3.1.2	Web Service Software Factory .....	34
3.2	Systemklassenbeschreibung .....	35
3.2.1	Auftragsserver.....	35
3.2.1.1	Service Contract.....	37
3.2.1.2	Data und Fault Contract.....	42
3.2.1.3	Host Model .....	42
3.2.1.4	DB Controller und Datenbanktabellen.....	43
3.2.1.5	Berechnungsserver .....	45
3.2.1.6	Logger .....	45
3.2.1.7	Auftragsabwicklung .....	45
3.2.1.8	SyncEvents .....	46
3.2.1.9	Einstellungen.....	46
3.2.1.10	Exception Handling .....	46
3.2.1.11	Ausblick.....	46
3.2.2	Berechnungsserver.....	47
3.2.2.1	Service Contract.....	48
3.2.2.2	Data- und Fault Contract.....	48
3.2.2.3	Logger .....	49
3.2.2.4	Einstellungen.....	49
3.2.2.5	Ausblick.....	49
3.2.3	Log Server.....	50
3.2.3.1	Service Contract.....	50
3.2.3.2	Data und Fault Contract.....	51
3.2.3.3	Datenbanktabelle .....	51
3.2.3.4	Einstellungen.....	51
3.2.3.5	Ausblick.....	51
3.2.4	Log Analyzer.....	52
3.2.4.1	Service Contract.....	52
3.2.4.2	Data Contact .....	52
3.2.4.3	Datenbanktabelle .....	52
3.2.4.4	Ausblick.....	52
3.2.5	Weboberfläche .....	53
3.2.5.1	Projektstruktur .....	55
3.2.5.2	Aufbau der Views .....	56
3.2.5.3	Zusätzlich verwendete Web Controls.....	56
3.2.5.4	Modul SoapCalc.....	57
3.2.5.5	Modul Administration.....	58
3.2.5.6	Errorhandling Webpage .....	58
3.2.5.7	Konfiguration .....	58
3.2.5.8	Ausblick.....	58
3.3	Systemklassen-Diagramm .....	59

---

3.4	Nachrichtenfolge .....	59
3.4.1	Weboberfläche – Auftragsserver .....	60
3.4.2	Auftragsserver – Logger .....	61
3.4.3	Auftragsserver - Berechnungsserver .....	62
3.5	Initialisierung der verschiedenen Objekte .....	63
3.6	Algorithmen .....	64
3.6.1	Lastindikator .....	64
3.6.1.1	Formel zur Berechnung des Lastindikators .....	64
3.6.1.2	Betrachtung verschiedener Zustände .....	64
4	Prototyp .....	65
4.1	WCF mit Web Service Software Factory .....	65
4.1.1	Absicht .....	65
4.1.2	Realisation .....	65
4.1.2.1	Service Contract .....	65
4.1.2.2	Data Contract .....	65
4.1.2.3	Host .....	66
4.1.3	Testergebnisse und Erkenntnisse .....	66
4.2	LINQ und die WCSF .....	67
4.2.1	Absicht .....	67
4.2.2	Realisation .....	67
4.2.3	Testergebnisse und Erkenntnisse .....	68
5	Testfälle .....	69
5.1	Auftragsserver .....	69
5.2	Berechnungsserver .....	73
5.3	Weboberfläche .....	74
5.4	Logger .....	77
5.5	Log Analyzer .....	77
6	Schlussbemerkungen .....	78
7	Anhang .....	79

# 1 Einleitung

## 1.1 Sinn und Zweck

Berechnungsaufgaben sollen auf verschiedene Server verteilt werden um die Verarbeitungszeit zu verkürzen und die Abläufe zu automatisieren. Die Software verwaltet diese Aufgaben und kontrolliert die Verteilung oder führt die Aufträge aus. Dem Benutzer steht eine Weboberfläche zur Verfügung, um den Dienst zu konfigurieren und Aufträge zu erfassen.

## 1.2 Umfang und Anwendungsbereich

SoapCalc umfasst folgende Komponenten

- Auftragsserver
- Berechnungsserver
- Log Server
- Weboberfläche als User Interface

SoapCalc wird im Umfeld der Mobilfunknetz- Planung und Optimierung eingesetzt. Aufträge sind z.B. die Berechnung der Abdeckung der Schweizer Bevölkerung mit dem Mobilfunkdienst HSPA oder die Verarbeitung von georeferenzierten Daten (Messungen, GIS Layer usw.). Grundsätzlich kann aber SoapCalc für beliebige Aufgaben eingesetzt werden, wo Berechnungen über die Kommandozeile oder per Skript gestartet werden können. So z.B. auch für die Umwandlung von WAV Dateien nach mp3.

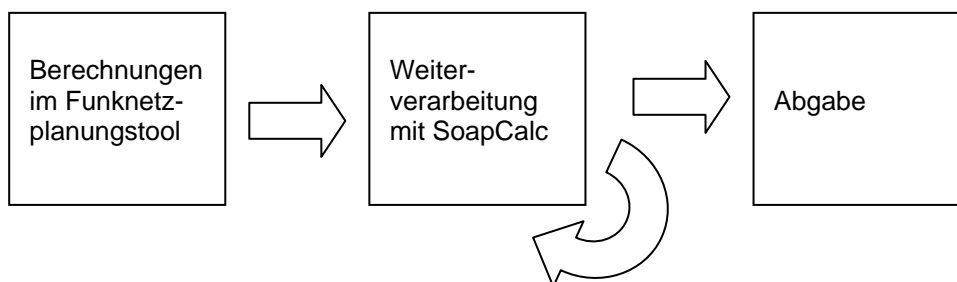


Abbildung 1: Anwendungsbereich von SoapCalc

## 1.3 Definitionen und Abkürzungen

### Auftragsserver

Der Auftragsserver verwaltet die Aufträge und verteilt diese an die Berechnungsserver.  
Siehe auch Kapitel 2.5.1.1

### Berechnungsserver

Ein Berechnungsserver nimmt die Aufträge entgegen und verarbeitet diese.  
Siehe auch Kapitel 2.5.1.2

### Log Server

Zentraler Log Server, der allen Diensten zur Verfügung steht.  
Siehe auch Kapitel 2.5.1.3

### Auftragsdefinition

Eine Auftragsdefinition definiert die Berechnung, welche ausgeführt werden soll.  
Siehe auch Kapitel 2.5.3.1

### Auftrag

Ein Auftrag beinhaltet alle benötigten Angaben und Daten, um von einem Berechnungsserver verarbeitet zu werden.  
Siehe auch Kapitel 2.5.3.2

**Auftragsvorlage**

Auftragsvorlagen sind Aufträge mit einer benannten Ablage. Sie sind wieder verwendbar.  
Siehe auch Kapitel 2.5.3.3

**Sammelauftrag**

Mehrere Auftragsvorlagen werden in einem Sammelauftrag zusammengefasst.  
Siehe auch Kapitel 2.5.3.4

**Auslastungsreport**

Der Auslastungsreport beinhaltet Informationen zur aktuellen Auslastung eines Berechnungsservers.  
Siehe auch Kapitel 2.5.3.5

**SoapCalc**

Projektname, setzt sich aus dem zu verwendenden Protokoll SOAP und Calc für berechnen zusammen.

## 1.4 Glossar

FTP - File Transfer Protocol  
GIS - Geographisches Informationssystem  
GSM - Global System for Mobile Communications  
HSPA - High Speed Packet Access  
IIS - Internet Information Server der Firma Microsoft (Web- und FTP- Server)  
LINQ - Language INtegrated Query  
SOA - Service Oriented Architecture  
SOAP - Simple Object Access Protocol  
UMTS - Universal Mobile Telecommunications System  
URI - Uniform Resource Identifier  
WCF - Windows Communication Foundation  
WCSF - Web Client Software Factory  
WSSF - Web Service Software Factory

## 1.5 Design- und Programmierkonvention

In dieser Arbeit sollen nach Möglichkeit die neuen Features von .NET 3.0 bzw. 3.5 eingesetzt werden. Konkret sind dies die WCF (.NET 3.0), LINQ (.NET 3.5) sowie die Neuerungen von ASP.NET. Anhand von Prototypen soll geprüft werden, ob diese Technologien im Rahmen dieser Arbeit eingesetzt werden können und Erfahrungen für zukünftige Projekte gesammelt werden.

Weiter sollen die Software Factories aus dem design & practice Umfeld von Microsoft verwendet werden, da diese bereits Funktionalität zur Verfügung stellen und auch in anderen Projekten der Inventis AG eingesetzt werden.

Dabei sind die Programmierrichtlinien der Inventis AG einzuhalten.

## 1.6 Abgrenzungen

Die Aufgabe umfasst die Entwicklung des Systems SoapCalc. Dies umfasst einen Auftragsserver, Berechnungsserver, einen Log Server und ein User Interface.  
Davon ausgenommen ist die Entwicklung der Skripts und Batches, welche die eigentlichen Aufträge darstellen.

Weiter muss das System nur innerhalb eines privaten Netzwerkes betrieben werden können und nicht zwingend über Netzwerkgrenzen hinweg (Firewallkonfiguration, Filetransfer).

Die Installation der Server kann manuell vorgenommen werden, d.h. es ist keine automatische Installation z.B. eines Berechnungsserverdienstes bei der ersten Anmeldung vorgesehen.

Alle Berechnungsserver werden als gleichwertig behandelt. Eine manuelle Auswahl des Berechnungsservers ist nicht möglich.

## 1.7 Zeitplan

KW	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
<b>Termine</b>																											
Beginn Diplomarbeit (10. März)																											
Kickoff Meeting (27. März)																											
Abgabe Pflichtenheft (24. April)																											
Kurs Präsentationstechnik (6. Mai)																											
Bestätigung vom Pflichtenheft (8. Mai)																											
Abgabe Diplomarbeit (11. Sept.)																											
<b>SoapCalc</b>																											
Systemanalyse																											
Pflichtenheft erstellen																											
Prototyping																											
Grobdesign																											
Detaildesign																											
Implementation																											
Test																											
Dokumentation																											
Ferien																											
Milestones						1					2								3								4
	Einarbeitung					Ausarbeitung					Konstruktion I												Konstruktion II				

### 1.7.1 Milestones

- Milestone 1: Die Grobanalyse des Systems ist abgeschlossen, das Pflichtenheft mit den Anforderungen erstellt
- Milestone 2: Die Prototypen des Systems funktionieren soweit, dass die Kommunikation zwischen den Diensten funktioniert. Ein Pseudoauftrag kann an einen Berechnungsserver verteilt werden, dieser wird verarbeitet. Die Aktionen werden geloggt.
- Milestone 3: Die Dienste Berechnungsserver und Logger sind abgeschlossen. Der Auftragsserver verwaltet Auftragsdefinitionen und Aufträge. Die Weboberfläche kann diese Funktionen nutzen.
- Milestone 4: Sammelaufträge können abgearbeitet werden. Präsentation vorbereiten, Projektabschluss.



## 2 Software Analyse

### 2.1 Grobübersicht mit MindMap

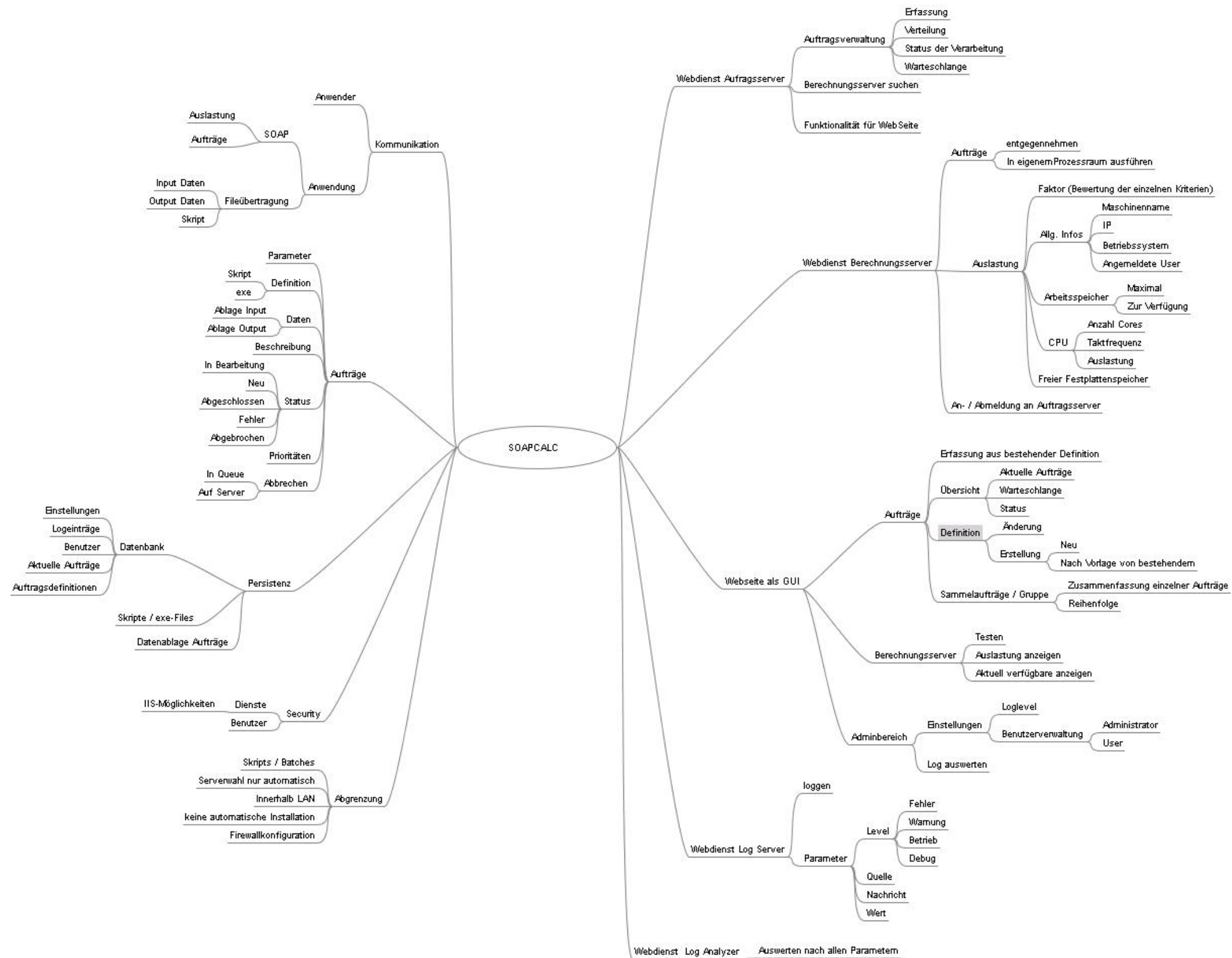


Abbildung 2: Mindmap SoapCalc

## 2.2 Schema

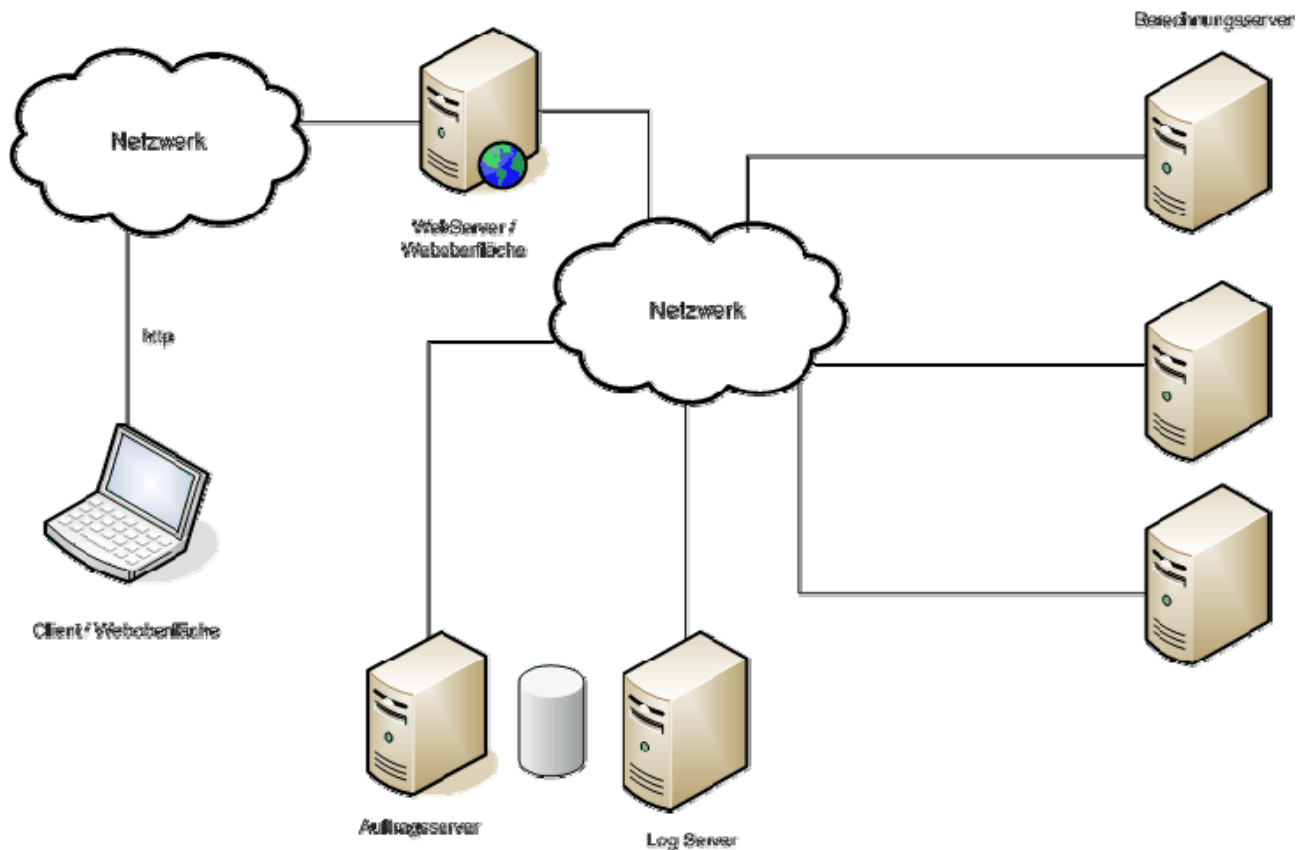


Abbildung 3: System - Schema

### 2.2.1 Auftragsserver

Der Auftragsserver stellt die Hauptkomponente des Systems dar. Hier werden die Aufträge verwaltet und an die Berechnungsserver verteilt. Auch wird die Funktionalität für die Weboberfläche zur Verfügung gestellt.

### 2.2.2 Berechnungsserver

Ein Berechnungsserver nimmt die Aufträge entgegen und verarbeitet diese. Die Anzahl der Berechnungsserver ist variabel (1..n).

### 2.2.3 Log Server

Ein zentraler Log Server, bestehend aus einem Logger und einem Log Analyzer. Der Logger steht allen Komponenten zur Verfügung.

### 2.2.4 User Interface / Weboberfläche

Dem Benutzer steht eine Weboberfläche zur Verfügung, um das System zu nutzen. Aufgeteilt ist diese Oberfläche in 3 Hauptteile:

- Aufträge, Auftragsdefinitionen und Sammelaufträge verwalten
- Übersicht Berechnungsserver
- Administrationsbereich (Passwortgeschützt)

## 2.3 Umfeld

### 2.3.1 Benutzer-Schnittstelle

Dem Benutzer steht eine Weboberfläche zur Verfügung, um das System zu nutzen. Diese kann mit einem Webbrowser angesprochen werden.

Die Benutzer des Systems sind technisch versiert (Fachspezialisten, Ingenieure).

### 2.3.2 Kommunikations-Schnittstelle

Als Kommunikationsprotokoll zwischen den Diensten wird das offene Protokoll SOAP verwendet. Dies wird von der WCF unterstützt.

Der Benutzer greift per http auf die Weboberfläche zu.

### 2.3.3 Software-Schnittstelle

Als Zielplattform stehen Server mit Windows Server 2003 und .NET 3.5 zur Verfügung. Auf den jeweiligen Servern ist IIS installiert. Der Auftragsserver und der LogServer verfügen zudem über eine SQL Express Installation.

### 2.3.4 Hardware

Als Hardware stehen fünf IBM xSeries 335 Server zur Verfügung. Diese sind mit jeweils 2 Dual Core Prozessoren und 4GB RAM ausgerüstet und sind über ein Fast Ethernet (100MBit/s) - Netzwerk miteinander verbunden.



Abbildung 4: Zur Verfügung stehende Hardware

Die Performance eines einzelnen vorhandenen physikalischen Servers reicht gut aus um einen Auftrags- und einen Log Server gleichzeitig zu betreiben.

Um eine optimale Performance für die Berechnungsserver zu erreichen, sollte auf einem physikalischen Server nur ein Berechnungsserver betrieben werden. Die vorhandenen Server sind für einen Berechnungsserver als Minimalkonfiguration zu betrachten.

## 2.4 Änderungen gegenüber dem Pflichtenheft

Gegenüber dem Pflichtenheft im Anhang A gab es im Verlauf der Arbeit folgende Änderungen

### 2.4.1 Anpassungen von Anforderungen

Anforderung Nr. 23

Komponente: Weboberfläche

Priorität: 1

Beschreibung: Mit der Weboberfläche kann man das System Administrieren

Änderung: Die Systemparameter des Auftragsservers werden mit der Weboberfläche nur dargestellt, das Ändern wird nicht benötigt.

Begründung: Es können nur Application Settings eingesetzt werden, welche zur Laufzeit nicht verändert werden können

Datum: 13.08.2008

Anforderung Nr. 24

Komponente: Weboberfläche

Priorität: 2

Beschreibung: Eine Benutzerverwaltung erlaubt das Administrieren der Benutzer.

Änderung: Die Benutzerverwaltung soll mit Active Directory vorgenommen werden.

Begründung: Active Directory steht bereits in der Umgebung zur Verfügung. Auch wird in weitere Applikationen der Firma die Benutzerverwaltung mit Active Directory vorgenommen

Datum: 13.08.2008

Anforderung Nr. 31

Komponente: Berechnungsserver

Priorität: 2

Beschreibung: Ein Berechnungsserver kann sich am Auftragsserver anmelden

Änderung: Ein Berechnungsserver meldet sich nicht am Auftragsserver an.

Begründung: Die Kontrolle über die eingesetzten Berechnungsserver soll nur vom Auftragsserver ausgehen

Datum: 13.08.2008

### 2.4.2 Nicht erfüllte Anforderungen

Aus Zeitgründen wurden zwei Anforderungen mit Priorität 3 nicht implementiert:

Anforderung Nr.	Beschreibung	Komponente	Priorität
33	Ein Berechnungsserver kann einen Auftrag ablehnen, wenn z.B. ein benötigtes Zusatzprogramm wie MapInfo nicht installiert ist.	Berechnungsserver	3
35	Der Berechnungsserver kann einen Performancefaktor berechnen.	Berechnungsserver	3

## 2.5 Spezifische Anforderungen

### 2.5.1 Funktionen

#### 2.5.1.1 Auftragsserver

Die Hauptkomponente von SoapCalc umfasst folgende Funktionalität:

##### **Auftragsabwicklung**

Der Auftragsserver nimmt neue Aufträge entgegen und speichert die Angaben in der Datenbank. Er kümmert sich um die Zusammenstellung der benötigten Daten, damit ein Auftrag in sich abgeschlossen an einen Berechnungsserver verteilt werden kann. Dazu erforderliche Auftragsdefinitionen werden ebenfalls vom Auftragsserver verwaltet. Anschliessend verteilt er die Aufträge an die zur Verfügung stehenden Berechnungsserver, unter Einhaltung der maximalen Auslastung. Abgeschlossene Aufträge nimmt er vom Berechnungsserver entgegen und informiert den Auftraggeber.

Zudem verwaltet er die Sammelaufträge und garantiert die Reihenfolge der einzelnen Aufträge.

##### **Verwaltung Berechnungsserver**

Die Verwaltung der Berechnungsserver wird durch den Auftragsserver gesteuert, dieser kann sich aber abmelden. Der Auftragsserver bezieht die vorhandenen Berechnungsserver automatisch in die Auftragsabwicklung ein. Dies beinhaltet auch das Abfragen von Auslastungsreports oder die Sperrung eines Berechnungsservers die durch den Benutzer veranlasst wird. Die Statusmeldungen vom Berechnungsserver werden entgegengenommen und entsprechend in die Datenbank eingetragen. Auch kann nach Berechnungsservern gesucht werden.

##### **Funktionalität Weboberfläche**

Die Inhalte der Weboberfläche werden auf dem Auftragsserver zusammengestellt. Dies sind die Angaben über die Aufträge, Auftragsdefinitionen und der Berechnungsserver.

Die Funktionalität der Weboberfläche wird an den Auftragsserver weitergeleitet, welcher diese anschliessend ausführt, wie z.B. das Abbrechen eines Auftrags.

##### **Datenbankzugriff**

Sämtliche Angaben zu Aufträgen, Auftragsdefinitionen, Sammelaufträgen und den Berechnungsservern werden in Tabellen der Datenbank abgelegt. Der Auftragsserver enthält Funktionen zur Verwaltung der einzelnen Business Objects (Einfügen, Ändern, Abfragen, Löschen), aber auch Funktionen welche für die Auftragsabwicklung benötigt werden wie z.B. die Abfrage des nächsten zu verteilenden Auftrags.

##### **File Store**

Die benötigten Dateien zu Aufträgen und Auftragsdefinitionen werden zentral auf dem Auftragsserver gespeichert. Sie stehen über eine Netzwerkfreigabe den übrigen Komponenten zur Verfügung.

#### 2.5.1.2 Berechnungsserver

Durch die Suche nach Berechnungsservern, welche vom Auftragsserver gesteuert wird, startet der Dienst des Berechnungsservers. Beim Beenden meldet sich dieser beim Auftragsserver ab.

Gestartet nimmt der Berechnungsserver Aufträge entgegen und führt diese aus. Beim Start wird die Prozess-ID des Prozesses, der den Auftrag ausführt, dem Auftragsserver zurückgegeben.

Aufträge können nur entgegengenommen werden, wenn der Berechnungsserver nicht schon zu stark ausgelastet ist. Dies ist abhängig von der Anzahl Cores und deren Auslastung sowie dem verfügbaren Arbeitsspeicher (hardwareabhängig). Ein Auslastungsreport (Siehe auch Kap. 2.5.3.6) kann erstellt werden, welcher diese Information beinhaltet.

Nach Abschluss des Auftrags wird eine Meldung an den Auftragsserver gesendet und die nicht mehr benötigten Daten gelöscht.

### 2.5.1.3 Log Server

Der Log Server besteht aus zwei Teilen:

#### Logger

Ein zentraler Logger steht allen Komponenten von SoapCalc zur Verfügung. Er nimmt einen Log Eintrag entgegen und schreibt diesen anschliessend in die Datenbank.

#### Log Analyzer

Mit dem Log Analyzer können die Log Einträge nach allen Informationen ausgewertet werden. Diese Funktionalität steht im Administrationsbereich der Weboberfläche zur Verfügung.

### 2.5.1.4 User Interface

Die Weboberfläche ist in drei Hauptteile aufgeteilt. Benutzer ohne Administratorenrechte haben Zugriff auf die Funktionalität der Aufträge und der Berechnungsserver. Die Gruppe der Administratoren zusätzlich auch auf die Funktionalität aus dem Administrationsbereich.

#### Aufträge verwalten

Auftragsdefinitionen können erstellt oder verändert werden. Aufträge und Auftragsvorlagen können aus bestehenden Auftragsdefinitionen erfasst werden.

Übersichten stellen die aktuellen Business Objekte und deren Status dar. Hier können diese bearbeitet werden.

Sammelaufträge können zusammengestellt werden.

#### Übersicht Berechnungsserver

Eine Übersicht stellt die aktuell angemeldeten Berechnungsserver mit dazugehöriger Information des Auslastungsreports dar.

Nicht angemeldete Berechnungsserver können gesucht werden.

#### Administrationsbereich (Passwortgeschützt)

Auf diesen Bereich haben nur berechtigte Benutzer Zugriff.

Die Konfiguration des Auftragsserver kann hier angezeigt werden (Verzeichnisse, Benachrichtigung, Timerintervalle usw.).

Die Auswertung des Logs mit dem Log Analyzer kann hier vorgenommen werden.

## 2.5.2 Mengengerüst

Für den Betrieb von SoapCalc werden folgende Komponenten benötigt:

- 1 Auftragsserver
- 1 Weboberfläche
- 1 Log Server
- 1..n Berechnungsserver

Zusammenstellung der aktuellen Situation pro Monat für die einzelnen Zyklen

Zyklus	reinen Rechenzeiten	Produzierte GIS Layer	Benötigte Aufträge
UMTS	3-4 Tage	ca. 100	ca. 300
GSM	2-3 Tage	ca. 80	ca. 240
Spezielle Aufträge	2-3 Tage	ca. 20	ca. 100

Da die Rechenzeit von der Anzahl Server abhängig ist, ergibt sich für die verbesserte Rechenzeit folgende Formel

$$Rechenzeit_{Neu} = Rechenzeit_{Alt} \cdot \frac{1}{(Server_{Anzahl} - 1)}$$

Die Anzahl der Aufträge pro Berechnungsserver ist hardwareabhängig.

Die Datenmenge einzelner Aufträge beträgt insgesamt bis ca. 100MB, welche auf mehrere Files verteilt sind.

## 2.5.3 Business Objects

Dieses Kapitel erläutert die Objekte, welche in mehreren Teilen des Systems verarbeitet werden. Diese sind als Data Contracts in den WCF Diensten und in den Tabellen der Datenbank abgebildet.

### 2.5.3.1 Auftragsdefinitionen

Eine Auftragsdefinition definiert den Berechnungstyp, welche ausgeführt werden soll. Dies beinhaltet

- Eindeutige ID (Guid)
- Name
- Skript / Batch / Executable
- Zusätzlich benötigte Files
- Liste der benötigten Parameter
- Beschreibung

Die Angaben der Auftragsdefinition werden in einer Datenbank abgespeichert. Pro Auftragsdefinition wird ein separates Verzeichnis angelegt. Für den Namen dieses Verzeichnisses wird die ID verwendet. Zusätzlich benötigte Daten wie Skripte oder benötigte statische Dateien usw. werden ebenfalls hier abgelegt und mit jedem Auftrag neu an den Berechnungsserver ausgeliefert.

### 2.5.3.2 Aufträge

Ein Auftrag wird aus einer Auftragsdefinition erstellt und beinhaltet alle benötigten Angaben, damit er an einen Berechnungsserver ausgeliefert und dort berechnet werden kann. Ein Auftrag ist in sich abgeschlossen.

Ein Auftrag ist wie folgt definiert:

- Eindeutige ID (Guid)
- Name
- Priorität
- Status
  - Neu
  - Bereit
  - Wartend
  - In Bearbeitung
  - Abgeschlossen
  - Fehler
  - Abgebrochen
  - Sammelauftrag
- Inputdaten (Files)
- Outputdaten (Files)
- ID der Auftragsdefinition
- Parameter für Skript
- ID des Sammelauftrags
- ID der Abhängigkeiten
- Berechnungsserver
- Prozess ID
- Start-, Verteil- und Endzeit

Die Angaben des Auftrags werden in der Datenbank gespeichert. Pro Auftrag wird ein separates Verzeichnis angelegt. Für den Namen dieses Verzeichnisses wird die ID verwendet. Neben einer Kopie der Auftragsdefinition werden die Inputdaten und nach der Berechnung die Outputdaten als komprimierte Datei im jeweiligen Verzeichnis abgelegt.

### 2.5.3.3 Auftragsvorlage

Sammelaufträge setzen sich aus Auftragsvorlagen zusammen. Diese beinhaltet dieselben Informationen wie ein Auftrag und werden auch in derselben Datenbanktabelle abgespeichert. Der Status ist dabei auf „Sammelauftrag“ gesetzt. Der Unterschied zwischen einem Auftrag und einer Auftragsvorlage ist nur im File Store, wo die Auftragsvorlagen in einem benannten Verzeichnis abgelegt werden. Auftragsvorlagen sind wieder verwendbar.

### 2.5.3.4 Sammelauftrag

Mehrere Auftragsvorlagen werden in einem Sammelauftrag zusammengefasst. Die Reihenfolge der einzelnen Aufträge wird dabei eingehalten. Die Aufträge können in Abhängigkeit eines anderen Auftrags sein. Ein einzelner Auftrag in einem Sammelauftrag kann maximal zwei vorgängige Aufträge als Startbedingung haben. Der Sammelauftrag wird auf dem Auftragsserver verarbeitet und als einzelne Aufträge an die Berechnungsserver weitergeleitet.

Folgende Angaben beschreiben den Sammelauftrag:

- Eindeutige ID Sammelauftrag (Guid)
- Name des Sammelauftrags
- Beschreibung
- Liste der Aufträge

Beispiel eines Sammelauftrags:

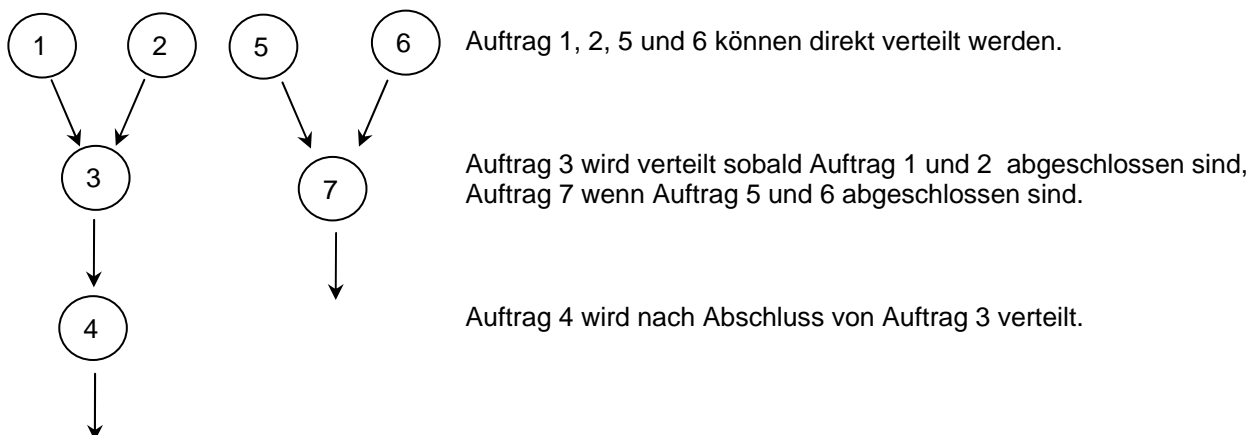


Abbildung 5: Beispiel eines Sammelauftrags

Die Angaben des Sammelauftrags werden in der Datenbank gespeichert.

### 2.5.3.5 Log Eintrag

Ein Log Eintrag beinhaltet folgende Information:

- Zeitstempel
- Sender
- Log-Level
  - Information
  - Debug
  - Warnung
  - Fehler
- Quelle
- Text/Daten

Log Einträge werden in der Datenbank gespeichert.



### 2.5.3.6 Auslastungsreport

Der Auslastungsreport gibt einen Überblick der aktuellen Auslastung eines Berechnungsservers. Er beinhaltet folgende Angaben:

- Allgemeine Infos
  - Maschinenname
  - Betriebssystem
  - IP
  - Anzahl eingeloggte Benutzer
- CPU
  - Anzahl Cores
  - Taktfrequenz
  - Auslastung
- Arbeitsspeicher
  - Maximal
  - Verfügbar
- Freier Speicherplatz Festplatte
- Flag Idle
- Flag Locked
- Berechnete Auslastung
- Version

Die Angaben der Berechnungsserver werden in der Datenbank gespeichert.

## 2.5.4 Datenbasis

### 2.5.4.1 Datenbank

Die Datenbank beinhaltet Tabellen für die Auftragsabwicklung zur Speicherung der Angaben der Aufträge, der Auftragsdefinitionen, der Sammelaufträge und der aktuellen Berechnungsserver sowie für den Logger.

Folgende Tabellen werden in der SoapCalc Datenbank benötigt:

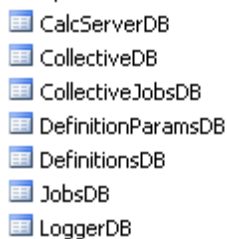


Abbildung 6: Tabellen der SoapCalc Datenbank

Tabellenname	Zweck
CalcServerDB	Enthält Informationen der aktuell angemeldeten Berechnungsserver
CollectiveDB	Enthält Angaben zu den Sammelaufträgen
CollectiveJobsDB	Enthält die Zuordnung der Auftragsvorlagen zum Sammelauftrag
DefinitionParamsDB	Enthält die Parameter der Auftragsdefinition
DefinitionsDB	Enthält die Angaben zu den Auftragsdefinitionen
JobsDB	Enthält die Angaben zu den Aufträgen
LoggerDB	Enthält die Log Einträge

### 2.5.4.2 File Store

Der File Store ist die Ordnerstruktur mit den Dateien, welche für die Auftragsabwicklung benötigt werden. Gespeichert werden die Dateien auf dem Auftragsserver. Die Daten sind über einen Netzwerk Share für den Web Server der Weboberfläche und die Berechnungsserver erreichbar.

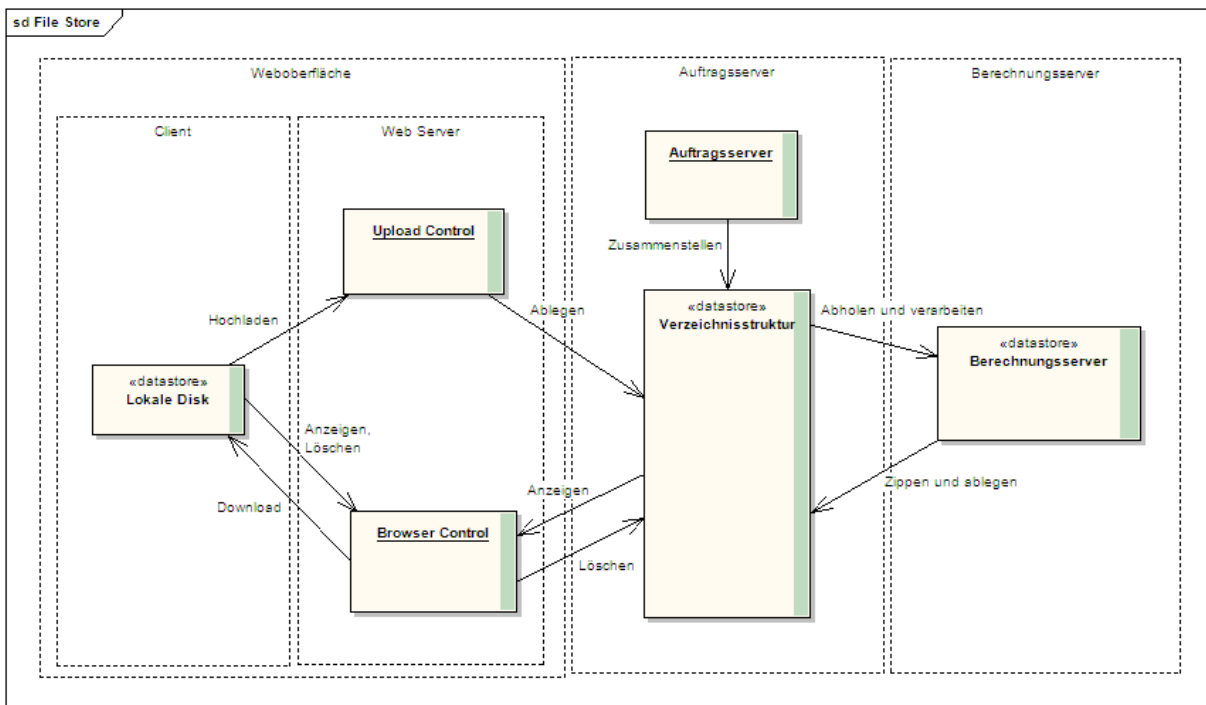


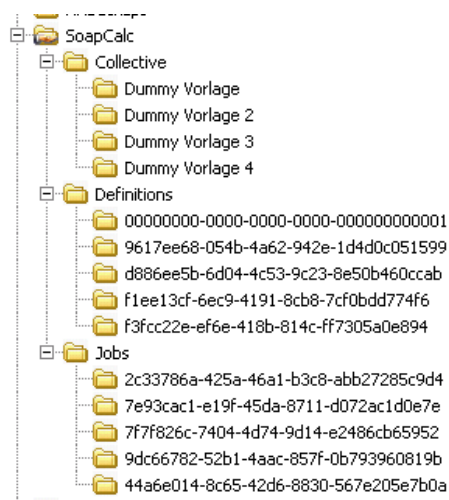
Abbildung 7: Übersicht File Handling

Um die Dateien verarbeiten zu können, werden diese mit Hilfe des Server Controls Upload auf dem Auftragsserver in der Verzeichnisstruktur abgelegt. Bei der Vorbereitung zur Verarbeitung werden die Dateien vom Auftragsserver zusammengestellt, die Dateien aus der Auftragsdefinition und diejenigen aus dem Auftrag werden zusammengeführt. Der Berechnungsserver holt diese Files nach Erhalt des Auftrags aus der Verzeichnisstruktur ab und verarbeitet diese.

Nach Abschluss der Verarbeitung werden die Resultate komprimiert und in der Verzeichnisstruktur des Auftragsservers abgelegt. Der Benutzer kann nun die Files über das Browser Control der Weboberfläche abholen.

Mit dem Browser Control hat der Benutzer zusätzlich die Möglichkeit, einzelne Dateien zu löschen.

Die Verzeichnisstruktur auf dem Auftragsserver ist wie folgt aufgebaut:



Der Rootpfad des Filestores wurde als Share freigegeben. Dieser beinhaltet pro Business Object einen Unterordner.

Abbildung 8: Ordnerstruktur des File Stores

## 2.5.5 Sicherheit

### 2.5.5.1 Komponenten

Für den Betrieb ist ein bereits geschütztes Umfeld vorgesehen. Damit werden nur Massnahmen für einen stabilen Betrieb benötigt, d.h. das nicht aus Versehen Funktionen von aussen angestossen werden können.

### 2.5.5.2 Benutzer

Es wird zwischen Benutzern und Administratoren unterschieden. Die Zugangsberechtigung wird in der Weboberfläche geprüft. Die Berechtigungen sind in Kapitel 2.5.1.4 beschrieben.

Es können mehrere Benutzer gleichzeitig eingeloggt sein.

## 2.5.6 Kommunikation

Als Kommunikationsplattform wird die Windows Communication Foundation (WCF) eingesetzt. Dies ist eine dienstorientierte Kommunikationsplattform für verteilte Anwendungen, wo viele Netzwerk-Funktionen zusammengeführt und standardisiert zur Verfügung gestellt werden. Durch die WCF werden die Kommunikationstechnologien DCOM, Enterprise Services, MSMQ, WSE und Web Services unter einer einheitlichen API zusammengestellt.

Das Hauptanwendungsgebiet von WCF liegt auf der Entwicklung Service-orientierter Architekturen. Die Windows Communication Foundation gehört zum .NET Framework 3.0.

Die WCF abstrahiert das Konzept des Endpunktes durch eine Trennung in Address, Binding und Contract (ABC-Prinzip).

### Address

Die Address ist ein URI, der den Ort des Dienstes beschreibt und somit seine Erreichbarkeit für die Dienstkonsumenten kennzeichnet.

### Binding

Ein Binding beschreibt die Art der Kommunikation. Dazu gehören Parameter wie Protokoll (HTTP, TCP, UDP und Windows-eigene Protokolle) und Codierung (binär, SOAP-Dokument, eigenes Format), sowie Sicherheitsaspekte (Verschlüsselung, Authentifizierung).

Das .NET Framework stellt vorgefertigte Bindungen für häufige Anwendungsfälle zur Verfügung. Diese können noch konfiguriert werden. Auch besteht die Möglichkeit eigene Bindings zu entwickeln.

### Contract

Ein Contract stellt eine Dienstdefinition dar. Contracts werden zur Entwicklungszeit als Interfaces (Schnittstellen) in einer beliebigen .NET-Sprache verfasst und zur Laufzeit durch die WCF in einen SOAP-Contract umgesetzt. Die Verwendung dieses Standards ist massgeblich für die Möglichkeiten des plattformunabhängigen Zugriffs auf Dienste möglich.

Alle Dienste von SoapCalc verwenden die WCF. Dabei werden alle als Singleton implementiert, als Host wird ein IIS verwendet. Zur Codierung wird SOAP verwendet. Als XML Namespace wurde <http://soapcalc.embe.ch> verwendet. Die Standard Parameter der TimeOut-Zeiten und die Nachrichtengrösse wurden an die Bedürfnisse der Anwendung angepasst.

## 2.6 Anwendungsfalldiagramme

### 2.6.1 Auftragsserver

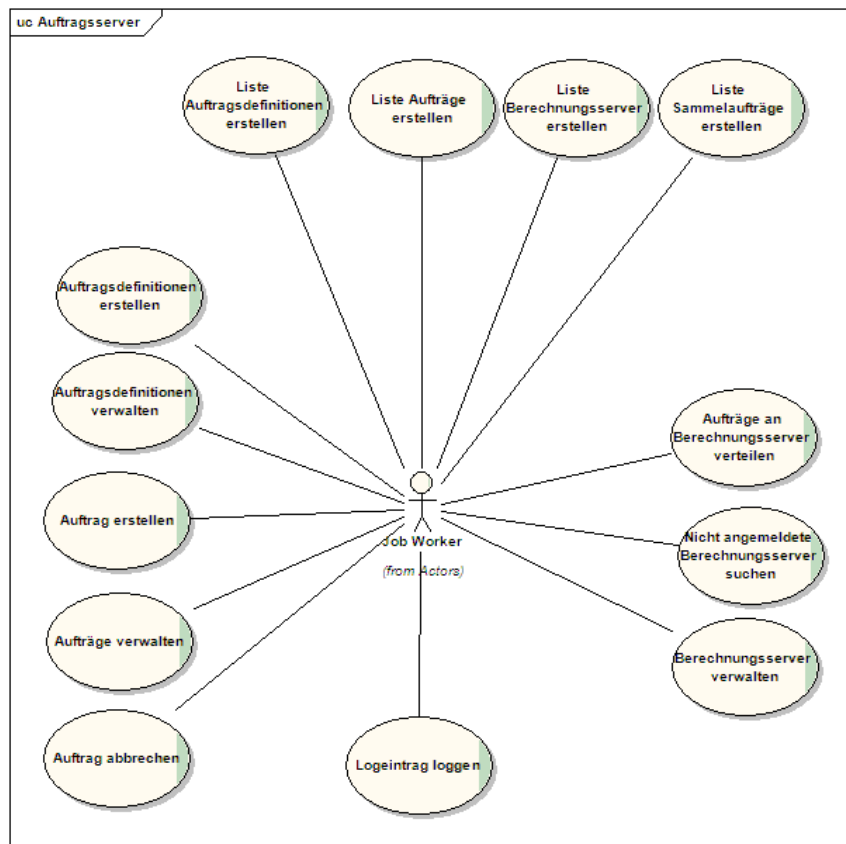


Abbildung 9: Anwendungsfälle Auftragsserver

### 2.6.2 Berechnungsserver

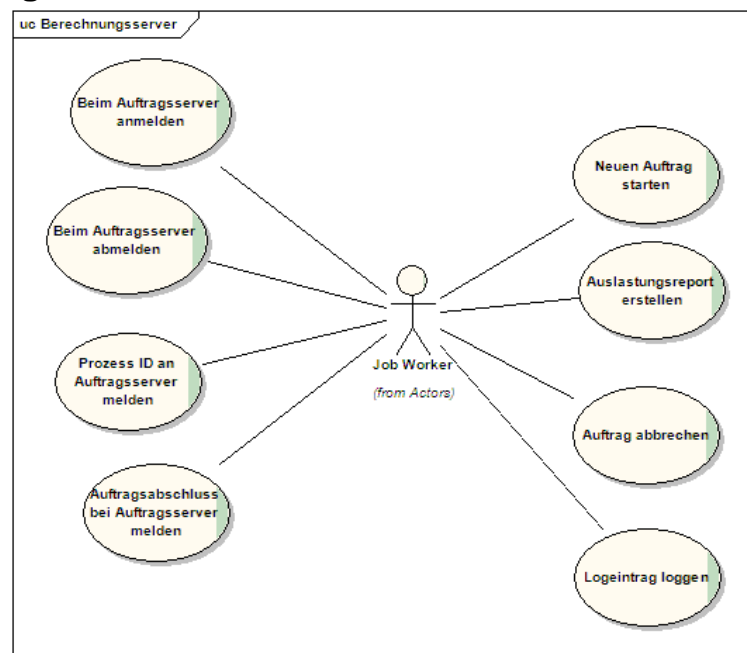


Abbildung 10: Anwendungsfälle Berechnungsserver

## 2.6.3 Log Server

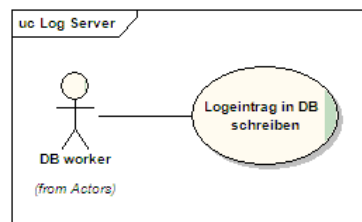


Abbildung 11: Anwendungsfall Log Server

## 2.6.4 Log Analyzer

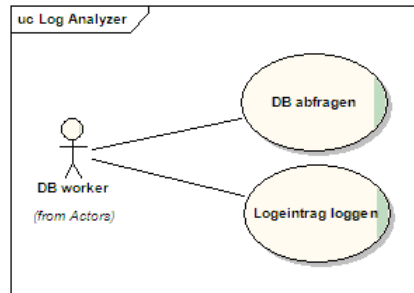


Abbildung 12: Anwendungsfälle Log Analyzer

## 2.6.5 Weboberfläche

### 2.6.5.1 Aufträge verwalten

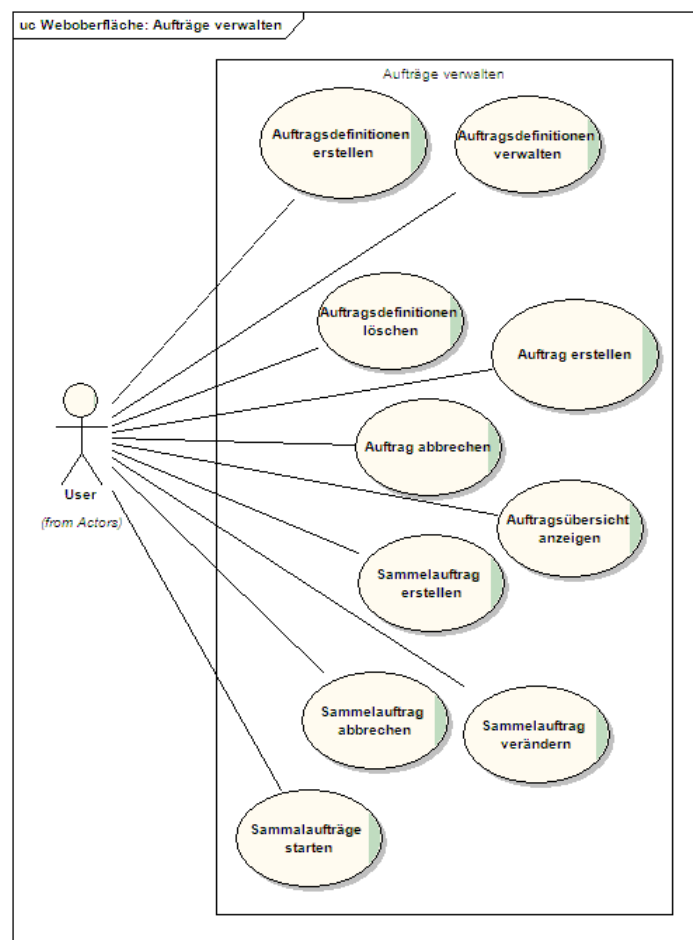


Abbildung 13: Anwendungsfälle Weboberfläche, Aufträge verwalten

### 2.6.5.2 Berechnungsserver

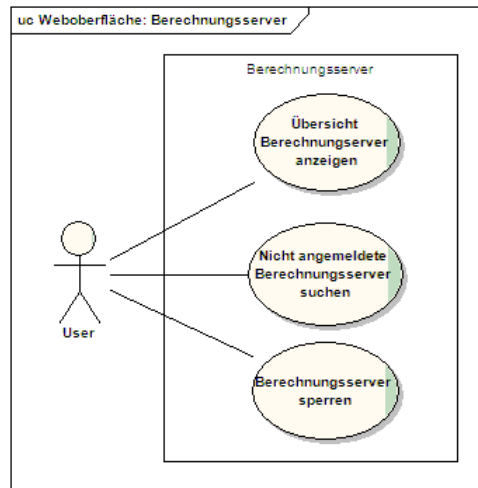


Abbildung 14: Anwendungsfälle Weboberfläche, Berechnungsserver

### 2.6.5.3 Administrationsbereich

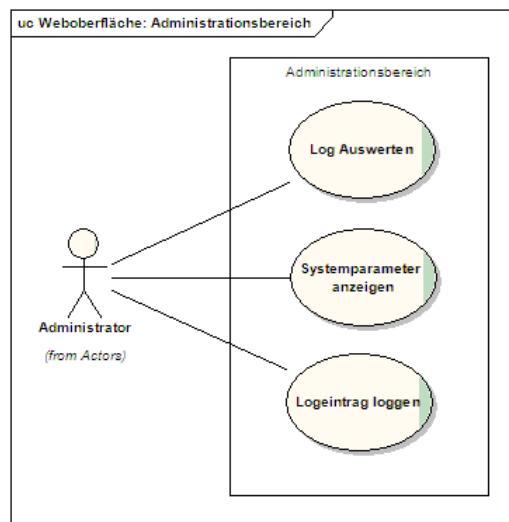


Abbildung 15: Anwendungsfälle Weboberfläche, Administrationsbereich

## 2.7 Anwendungsfallbeschreibungen

Anwendungsfall:	<b>Auftrag abbrechen</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	09.06.2008
Beschreibung:	Abbrechen eines bestehenden Auftrags. Dieser kann noch in der Warteschlange oder bereits bei der Berechnung sein. Im Falle der Berechnung müssen die nicht mehr benötigten Daten auf dem Berechnungsserver gelöscht werden.		
Akteure	Benutzer, SoapMain Worker		
Nachbedingung	Auftrag abgebrochen, Daten auf Berechnungsserver bereinigt		
Vorbedingung	Auftrag gestartet und noch nicht beendet		
Eingehende Daten	Guid des Auftrags		
Ausgehende Daten			
Bemerkungen			
Anforderung Nr.	14, 14.1, 14.2, 14.3		

Anwendungsfall:	<b>Auftrag erstellen</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	05.05.2008
Beschreibung:	Erstellen eines Auftrags mit der Vorlage einer Auftragsdefinition. Benötigt werden zusätzlich folgende Angaben: <ul style="list-style-type: none"> <li>• Name</li> <li>• Priorität</li> <li>• Status (In Bearbeitung, Neu, Abgeschlossen, Fehler, Abgebrochen)</li> <li>• Inputdaten</li> <li>• Outputdaten</li> </ul>		
Akteure	Benutzer, SoapMain Worker		
Nachbedingung	Angaben in DB gespeichert		
Vorbedingung	Auftragsdefinition vorhanden		
Eingehende Daten	Angaben zu Auftrag, benötigte Files		
Ausgehende Daten	Berechnete Daten gemäss Auftrag		
Bemerkungen			
Anforderung Nr.	10.1, 10.3, 10.4		

Anwendungsfall:	<b>Auftragsabschluss bei Auftragsserver melden</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	23.05.2008
Beschreibung:	Nach Abschluss des erteilten Auftrags eine Meldung an den Auftragsserver senden.		
Akteure	SoapCalc Worker		
Nachbedingung	ExitCode an Auftragsserver gemeldet, Dateien im Output Ordner abgelegt		
Vorbedingung	Auftrag auf Berechnungsserver gestartet		
Eingehende Daten	ExitCode		
Ausgehende Daten			
Bemerkungen			
Anforderung Nr.	30.4		

Anwendungsfall:	<b>Auftragsdefinitionen erstellen</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	05.05.2008
Beschreibung:	Erstellt eine neue Auftragsdefinition und speichert diese ab. Benötigt werden die Angaben: <ul style="list-style-type: none"> <li>• Name</li> <li>• Skript / Batch / Executable</li> <li>• Parameter</li> <li>• Beschreibung</li> </ul>		
Akteure	Benutzer, SoapMain Worker		
Nachbedingung	Angaben in DB gespeichert		
Vorbedingung			
Eingehende Daten	Angaben zur Auftragsdefinition, Benötigte Dateien		
Ausgehende Daten			
Bemerkungen			
Anforderung Nr.	11.1, 11.3, 11.4		

Anwendungsfall:	<b>Auftragsdefinitionen löschen</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	05.05.2008
Beschreibung:	Löschen einer bestehenden Auftragsdefinition, welche in keinem Auftrag mehr eingesetzt wird.		
Akteure	Benutzer, SoapMain Worker		
Nachbedingung	DB Eintrag und Dateien gelöscht		
Vorbedingung	Auftragsdefinition vorhanden		
Eingehende Daten	Guid		
Ausgehende Daten			
Bemerkungen			
Anforderung Nr.	11.6		

Anwendungsfall:	<b>Auftragsdefinitionen bearbeiten</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	05.05.2008
Beschreibung:	Bearbeiten einer bestehenden Auftragsdefinition		
Akteure	Benutzer, SoapMain Worker		
Nachbedingung	Auftragsdefinition auf dem gewünschten Stand		
Vorbedingung	Auftragsdefinition vorhanden		
Eingehende Daten	Angaben zur Auftragsdefinition, Benötigte Dateien		
Ausgehende Daten			
Bemerkungen			
Anforderung Nr.	11, 11.2, 11.6		



Anwendungsfall:	<b>Auftragsübersicht anzeigen</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	05.05.2008
Beschreibung:	Anzeigen einer Übersicht der aktuellen Aufträge		
Akteure	Benutzer, SoapMain Worker		
Nachbedingung	Auftragsübersicht wird angezeigt		
Vorbedingung	Benutzer fordert Übersicht an		
Eingehende Daten			
Ausgehende Daten	Liste mit Auftragsübersicht		
Bemerkungen			
Anforderung Nr.	10.5, 21		

Anwendungsfall:	<b>Aufträge an Berechnungsserver verteilen</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	05.05.2008
Beschreibung:	Verteilt die Aufträge an die Berechnungsserver. Dabei wird die Auslastung der Berechnungsserver berücksichtigt. Wenn alle Berechnungsserver maximal ausgelastet sind bleiben die Aufträge in der Warteschlange.		
Akteure	SoapMain Worker		
Nachbedingung	Auftrag an Berechnungsserver verteilt		
Vorbedingung	Auftrag mit Status Bereit, Berechnungsserver nicht gesperrt und verfügbar		
Eingehende Daten	Auftrag		
Ausgehende Daten			
Bemerkungen			
Anforderung Nr.	13, 13.1, 13.2, 13.3		

Anwendungsfall:	<b>Aufträge verwalten</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	05.05.2008
Beschreibung:	Verwaltet die aktuellen Aufträge (Ändern, Löschen)		
Akteure	Benutzer, SoapMain Worker		
Nachbedingung	Auftrag auf dem gewünschten Stand		
Vorbedingung	Auftrag vorhanden		
Eingehende Daten	Angaben zum Auftrag, Benötigte Dateien		
Ausgehende Daten			
Bemerkungen			
Anforderung Nr.	10, 10.2, 10.3, 10.4, 10.6, 12, 12.3		

Anwendungsfall:	<b>Auslastungsreport erstellen</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	05.05.2008
Beschreibung:	Zusammenstellen eines aktuellen Auslastungsreports. Dieser beinhaltet <ul style="list-style-type: none"> <li>• Maschinename</li> <li>• Betriebssystem</li> <li>• IP</li> <li>• Anzahl Cores</li> <li>• CPU Taktfrequenz</li> <li>• CPU Auslastung</li> <li>• Freier Speicherplatz</li> <li>• Maximaler Arbeitsspeicher</li> <li>• Verfügbarer Arbeitsspeicher</li> <li>• Version</li> </ul>		
Akteure	SoapCalc Worker		
Nachbedingung	Auslastungsreport erstellt		
Vorbedingung	Anfrage vom Auftragsserver		
Eingehende Daten			
Ausgehende Daten	Auslastungsreport		
Bemerkungen			
Anforderung Nr.	34, 35		

Anwendungsfall:	<b>Beim Auftragsserver abmelden</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	05.05.2008
Beschreibung:	Abmeldung beim Auftragsserver beim Herunterfahren des Berechnungsservers.		
Akteure	SoapCalc Worker		
Nachbedingung	Berechnungsserver beim Auftragsserver abgemeldet		
Vorbedingung	Berechnungsserver beim Auftragsserver angemeldet		
Eingehende Daten			
Ausgehende Daten	Auslastungsreport		
Bemerkungen			
Anforderung Nr.	32		

Anwendungsfall:	<b>Berechnungsserver verwalten</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	05.05.2008
Beschreibung:	Führt eine Liste von allen angemeldeten Berechnungsservern und hält die Auslastungsreports der jeweiligen Server aktuell. Beim Start wird nach laufenden Berechnungsservern gesucht.		
Akteure	SoapMain Worker		
Nachbedingung			
Vorbedingung			
Eingehende Daten	Änderungen der Berechnungsserver (Abmeldungen)		
Ausgehende Daten			
Bemerkungen			
Anforderung Nr.	15, 15.2		

Anwendungsfall:	<b>DB abfragen</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	23.05.2008
Beschreibung:	Abfragen der Datenbank für die Auftragsabwicklung		
Akteure	Benutzer, SoapMain Worker		
Nachbedingung	Angeforderte Angaben stehen zur Verfügung		
Vorbedingung			
Eingehende Daten	Filterkriterien		
Ausgehende Daten	Angeforderte Angaben		
Bemerkungen			
Anforderung Nr.	10		

Anwendungsfall:	<b>Liste Auftragsdefinitionen erstellen</b>		
Erstellt am:	05.05.2008	Bearbeitet am:	05.05.2008
Beschreibung:	Erstellt eine Liste der Auftragsdefinitionen.		
Akteure	Benutzer, SoapMain Worker		
Nachbedingung			
Vorbedingung	Benutzer fordert Übersicht an		
Eingehende Daten			
Ausgehende Daten	Liste mit Auftragsdefinitionen erstellt		
Bemerkungen			
Anforderung Nr.	11.5		

Anwendungsfall:	<b>Liste Aufträge erstellen</b>		
Erstellt am:	05.05.2008	Bearbeitet am:	05.05.2008
Beschreibung:	Erstellt eine Liste der aktuellen Aufträge		
Akteure	Benutzer, SoapMain Worker		
Nachbedingung			
Vorbedingung	Benutzer fordert Übersicht an		
Eingehende Daten			
Ausgehende Daten	Liste mit Aufträgen erstellt		
Bemerkungen			
Anforderung Nr.	10.5		

Anwendungsfall:	<b>Liste Berechnungsserver erstellen</b>		
Erstellt am:	05.05.2008	Bearbeitet am:	05.05.2008
Beschreibung:	Erstellt eine Liste der aktuell angemeldeten Berechnungsserver		
Akteure	Benutzer, SoapMain Worker		
Nachbedingung			
Vorbedingung	Benutzer fordert Übersicht an		
Eingehende Daten			
Ausgehende Daten	Liste der Berechnungsserver erstellt		
Bemerkungen			
Anforderung Nr.	15.1		

Anwendungsfall:	<b>Liste Sammelaufträge erstellen</b>		
Erstellt am:	05.05.2008	Bearbeitet am:	05.05.2008
Beschreibung:	Eine Liste der aktuellen Sammelaufträge kann erstellt werden.		
Akteure	Benutzer, SoapMain Worker		
Nachbedingung			
Vorbedingung	Benutzer fordert Übersicht an		
Eingehende Daten			
Ausgehende Daten	Liste der Sammelaufträge erstellt		
Bemerkungen			
Anforderung Nr.	12		

Anwendungsfall:	<b>Log Auswerten</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	05.05.2008
Beschreibung:	Abfragen der Datenbank nach Log Einträgen mit Angabe der Filterkriterien.		
Akteure	Benutzer, Log Analyzer Worker		
Nachbedingung	Anzeigen der gefilterten Log Einträge		
Vorbedingung			
Eingehende Daten	Filterkriterien		
Ausgehende Daten	Liste der Log Einträge		
Bemerkungen			
Anforderung Nr.	50		

Anwendungsfall:	<b>Logeintrag in DB schreiben</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	09.06.2008
Beschreibung:	Eintragen der Log Angaben in die Datenbank.		
Akteure	SoapLogger Worker		
Nachbedingung	Log Eintrag in DB gespeichert		
Vorbedingung	Log Eintrag in Queue		
Eingehende Daten	Log Eintrag		
Ausgehende Daten			
Bemerkungen			
Anforderung Nr.	40, 41, 42, 60		

Anwendungsfall:	<b>Logeintrag loggen</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	09.06.2008
Beschreibung:	Schreibt eine Meldung an den Log Server. Als Information mitgegeben werden <ul style="list-style-type: none"> <li>• Zeitstempel</li> <li>• Sender</li> <li>• Log-Level (Fehler, Warnung, Betrieb, Debug)</li> <li>• Quelle</li> <li>• Text/Daten</li> </ul>		
Akteure	Benutzer, SoapMain Worker, SoapCalc Worker		
Nachbedingung	Log Eintrag in Queue		
Vorbedingung			
Eingehende Daten	Log Eintrag		
Ausgehende Daten			
Bemerkungen			
Anforderung Nr.	40		

Anwendungsfall:	<b>Neuen Auftrag starten</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	23.05.2008
Beschreibung:	Entgegen nehmen und Starten eines neuen Auftrags mit den mitgegebenen Parametern und Dateien.		
Akteure	SoapMain Worker, SoapCalc Worker		
Nachbedingung	Berechnung gestartet		
Vorbedingung	Neuer Auftrag vom Auftragsserver entgegen genommen		
Eingehende Daten	Auftrag		
Ausgehende Daten	Berechnete Daten		
Bemerkungen			
Anforderung Nr.	30, 30.1, 30.2, 33		

Anwendungsfall:	<b>Nicht angemeldete Berechnungsserver suchen</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	05.05.2008
Beschreibung:	Suche nach nicht angemeldeten aber aktiven Berechnungsservern.		
Akteure	Benutzer, SoapMain Worker		
Nachbedingung	Neu gefundene Berechnungsserver eingetragen		
Vorbedingung			
Eingehende Daten			
Ausgehende Daten			
Bemerkungen			
Anforderung Nr.	17		

Anwendungsfall:	<b>Prozess ID an Auftragsserver melden</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	23.05.2008
Beschreibung:	Meldet die ID des Prozesses und den Servernamen an den Auftragsserver für den Fall das die Berechnung abgebrochen werden muss.		
Akteure	SoapCalc Worker		
Nachbedingung	Prozess ID und Servername an Auftragsserver gemeldet		
Vorbedingung	Auftrag in Berechnung		
Eingehende Daten			
Ausgehende Daten	Process ID und Servername		
Bemerkungen			
Anforderung Nr.	30.3		

Anwendungsfall:	<b>Sammelauftrag abbrechen</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	05.05.2008
Beschreibung:	Abbrechen eines bestehenden Sammelauftrags, bzw. aller einzelnen Aufträge		
Akteure	Benutzer, SoapMain Worker		
Nachbedingung	Sammelauftrag abgebrochen, inkl. der einzelnen Aufträge		
Vorbedingung	Sammelauftrag aufgegeben		
Eingehende Daten	Guid des Sammelauftrags		
Ausgehende Daten			
Bemerkungen			
Anforderung Nr.	14, 14.1, 14.2, 14.3		

Anwendungsfall:	<b>Sammelauftrag erstellen</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	05.05.2008
Beschreibung:	Erstellen eines Sammelauftrags durch Zusammensetzen von Auftragsvorlagen. Angabe der Abhängigkeit zwischen den Aufträgen.		
Akteure	Benutzer, SoapMain Worker		
Nachbedingung	Sammelauftrag wie der Benutzer diesen zusammengestellt hat wurde abgespeichert		
Vorbedingung	Aufträge für Sammelauftrag vorbereitet		
Eingehende Daten	Aufträge		
Ausgehende Daten			
Bemerkungen			
Anforderung Nr.	12.1		

Anwendungsfall:	<b>Sammelauftrag verändern</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	05.05.2008
Beschreibung:	Bearbeiten eines bestehenden Sammelauftrags		
Akteure	Benutzer, SoapMain Worker		
Nachbedingung	Sammelauftrag wie der Benutzer in zusammen gestellt hat wurde abgespeichert		
Vorbedingung	Sammelauftrag vorhanden		
Eingehende Daten	Änderungen des Benutzers		
Ausgehende Daten			
Bemerkungen			
Anforderung Nr.	12.2		

Anwendungsfall:	<b>Sammelauftrag starten</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	05.05.2008
Beschreibung:	Starten eines bestehenden Sammelauftrags		
Akteure	Benutzer, SoapMain Worker		
Nachbedingung	Die Auftragsvorlagen des Sammelauftrags wie der Benutzer diesen zusammengestellt hat werden dupliziert und als neue Aufträge aufgeben		
Vorbedingung	Sammelauftrag vorhanden		
Eingehende Daten	Sammelauftrag		
Ausgehende Daten			
Bemerkungen			
Anforderung Nr.	12		

Anwendungsfall:	<b>Systemparameter anzeigen</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	09.06.2008
Beschreibung:	Anzeigen der Konfigurationsparameter für den Auftragsserver wie z.B. die Verzeichnisse, Benachrichtigung, Timerintervalle usw.		
Akteure	Administrator		
Nachbedingung	Liste der Konfigurationsparameter dargestellt		
Vorbedingung	Übersicht vom Benutzer angefordert		
Eingehende Daten			
Ausgehende Daten			
Bemerkungen			
Anforderung Nr.	23		

---

Anwendungsfall:	<b>Übersicht Berechnungserver anzeigen</b>		
Erstellt am:	17.04.2008	Bearbeitet am:	05.05.2008
Beschreibung:	Anzeigen einer Übersicht aller angemeldeten Berechnungserver		
Akteure	Benutzer, SoapMain Worker		
Nachbedingung	Liste der Berechnungsserver dargestellt		
Vorbedingung	Übersicht vom Benutzer angefordert		
Eingehende Daten			
Ausgehende Daten	Liste der Berechnungsserver		
Bemerkungen			
Anforderung Nr.	22		

## 2.8 Aktivitätsdiagramme

### 2.8.1 Auftragsserver

#### 2.8.1.1 Neuer Job

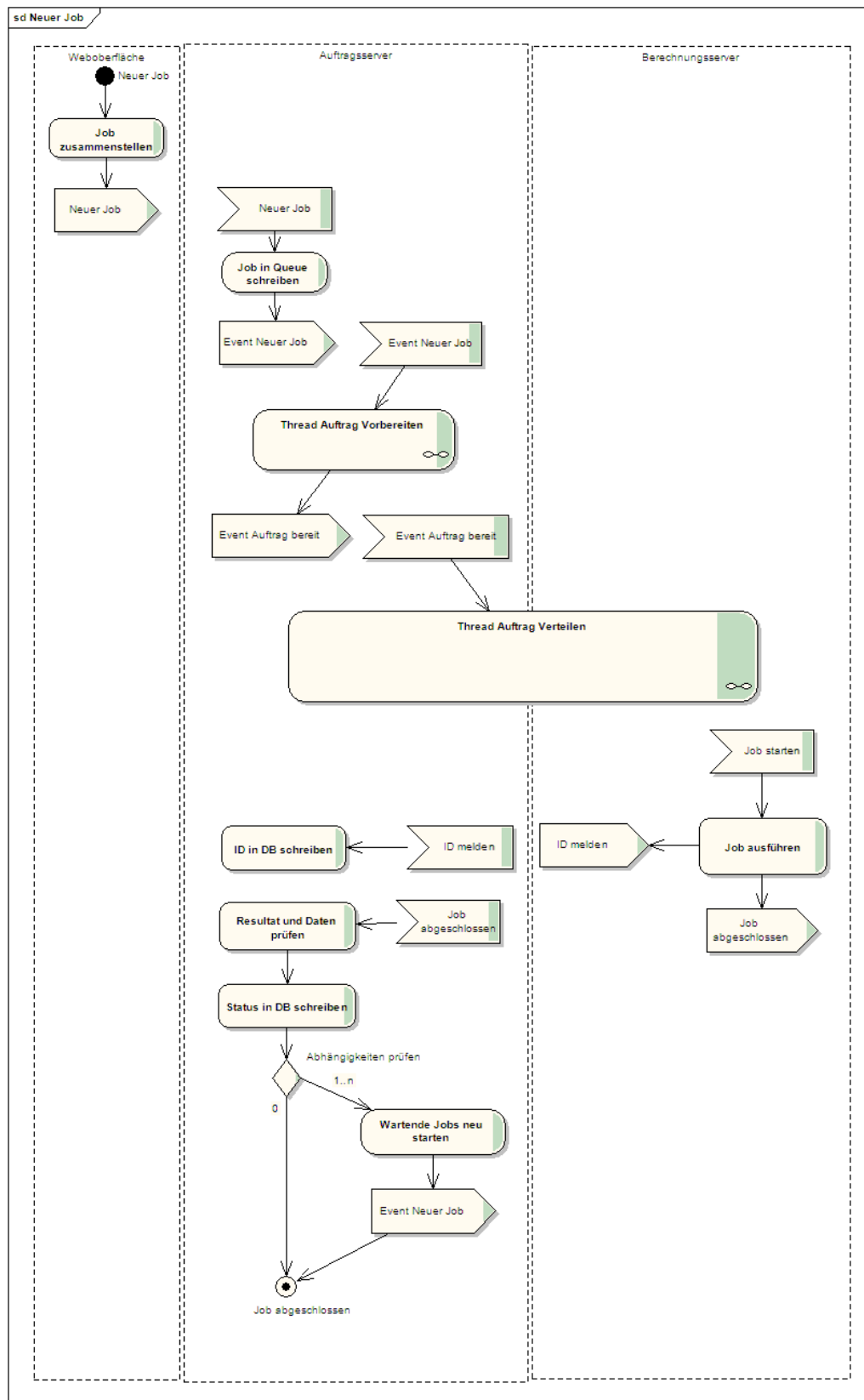


Abbildung 16: Neuer Job



### 2.8.1.2 Auftrag vorbereiten

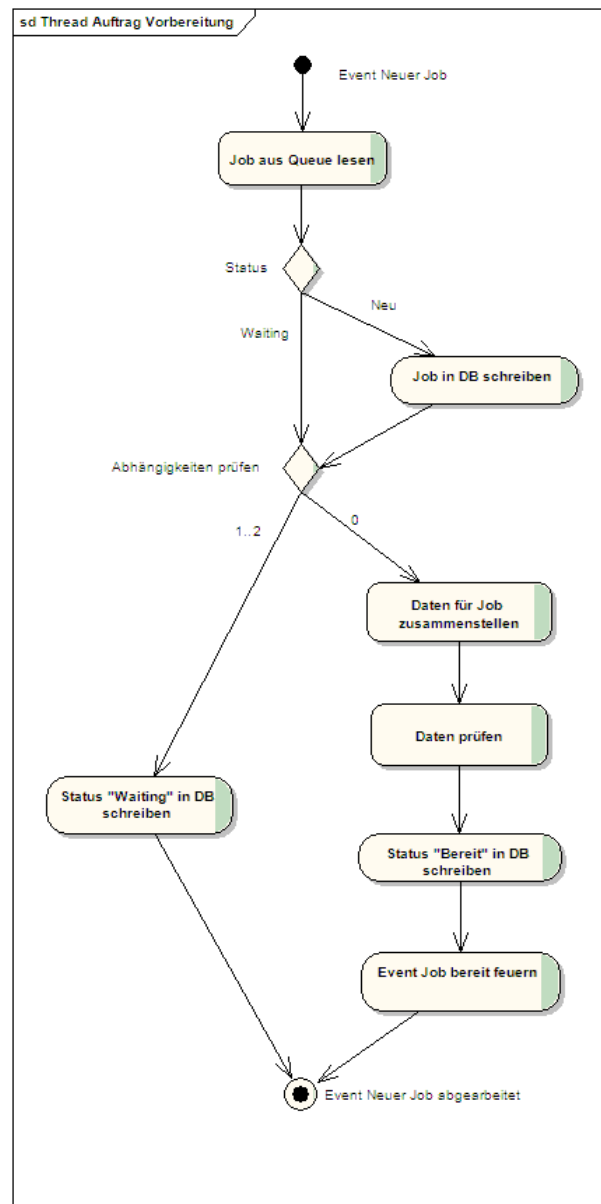


Abbildung 17: Ablauf Auftrag vorbereiten

### 2.8.1.3 Auftrag verteilen

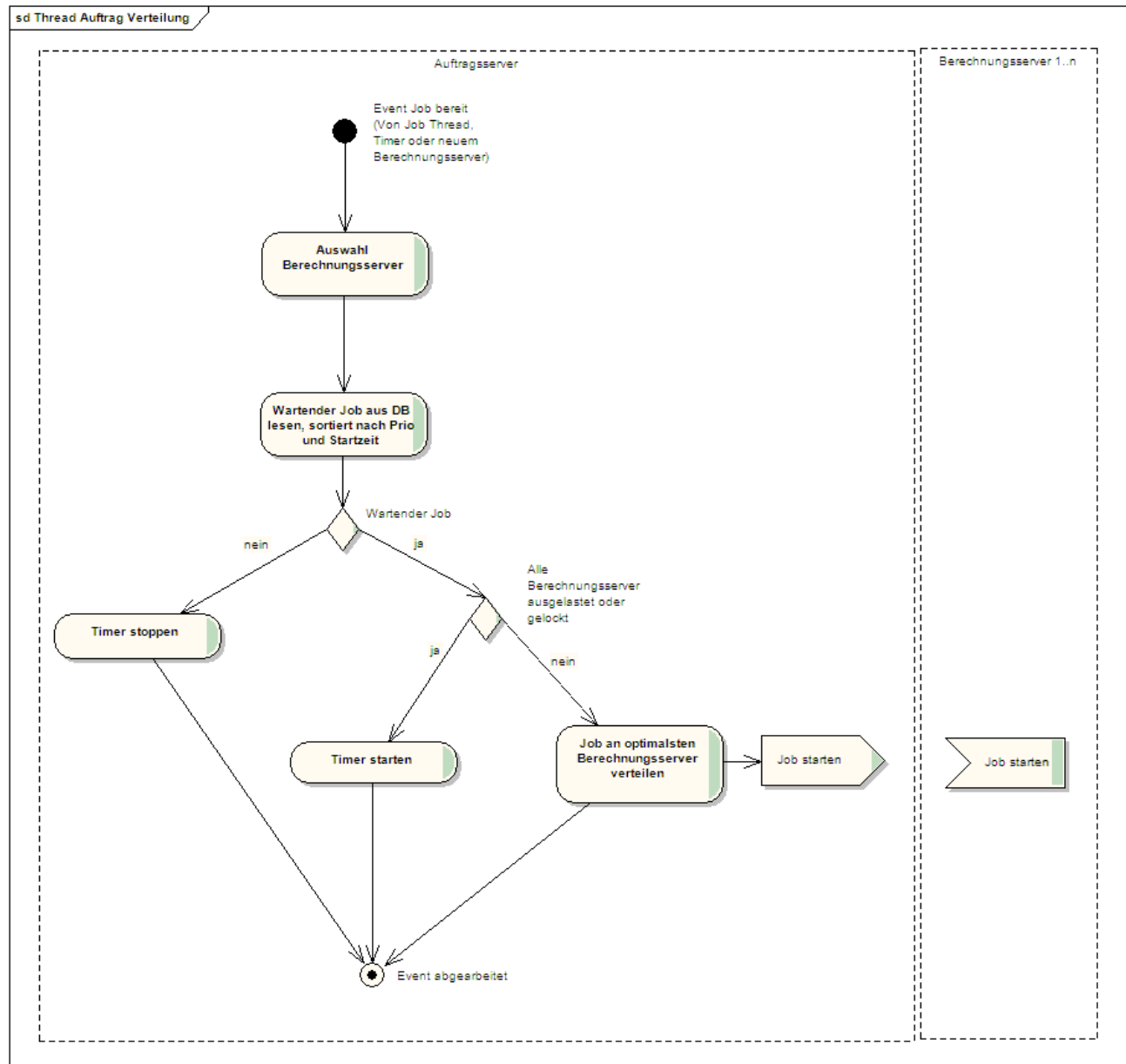


Abbildung 18: Ablauf Auftrag verteilen

### 2.8.1.4 Auswahl Berechnungsserver

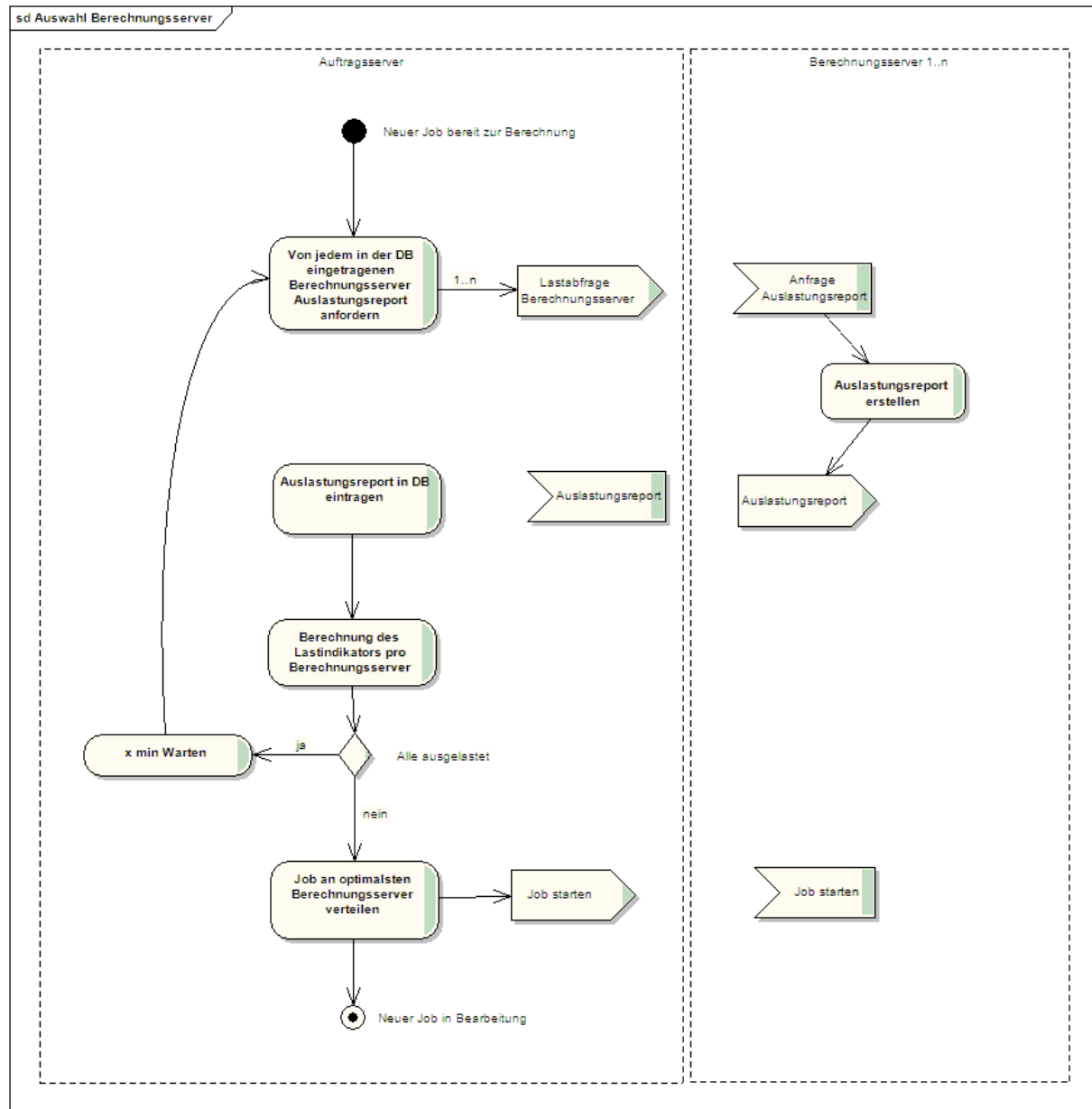


Abbildung 19: Auswahl der Berechnungsserver

## 2.8.2 Berechnungsserver

### 2.8.2.1 Abmeldung Berechnungsserver vom Auftragsserver

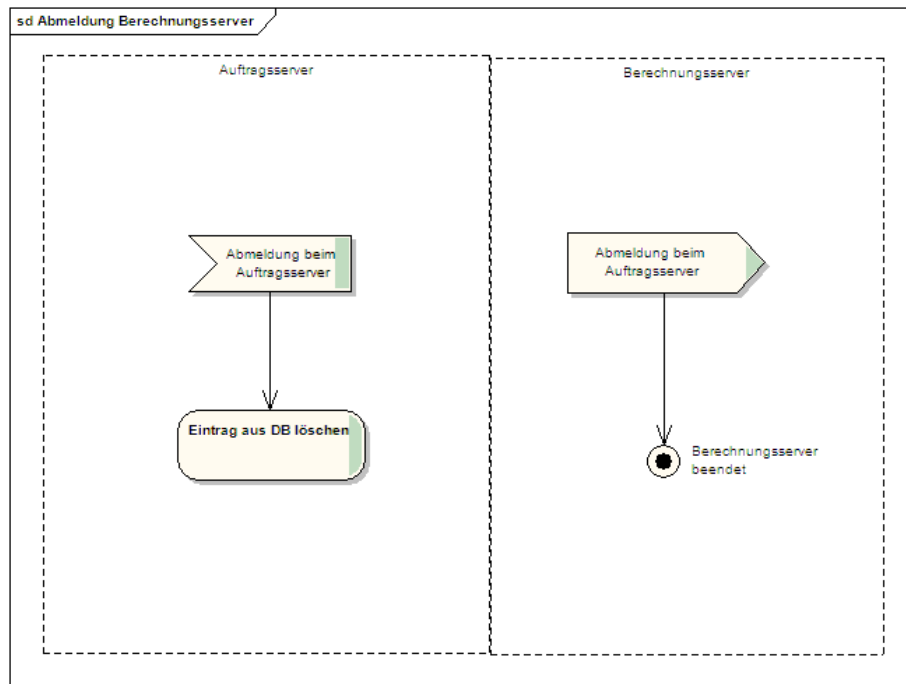


Abbildung 20: Abmeldung Berechnungsserver vom Auftragsserver

## 2.8.3 Log Server

### 2.8.3.1 Neuer Log Eintrag

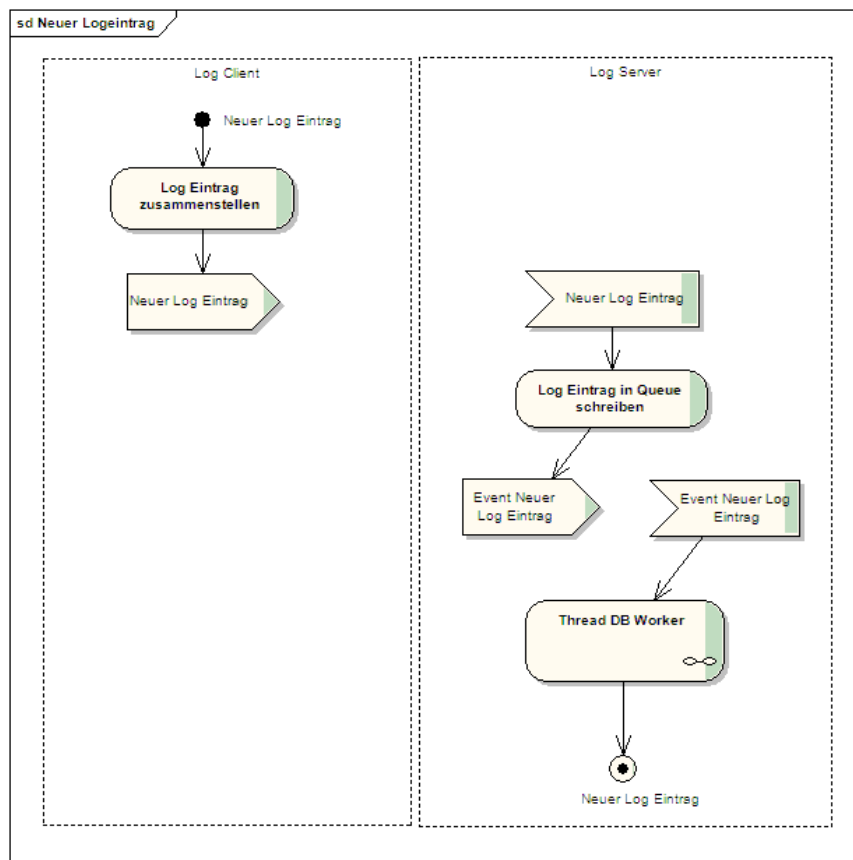


Abbildung 21: Ablauf Neuer Log Eintrag

### 2.8.3.2 Eintrag in DB speichern

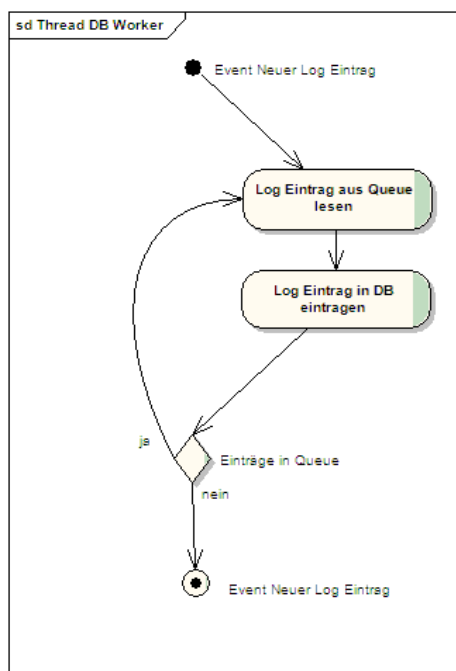


Abbildung 22: Log Eintrag in DB speichern

## 3 Software Design

### 3.1 Einleitung in die Software Factories

Als Designunterstützung werden zwei Software Factories aus dem design & practice Umfeld von Microsoft eingesetzt. Diese integrieren sich nahtlos ins Visual Studio. Die Systemklassen werden direkt mit Hilfe dieser Software Factories generiert.

Anhand der Prototypen wurde diese für den Einsatz in diesem Projekt überprüft. Detailliertere Informationen siehe auch Kapitel 4.

#### 3.1.1 Web Client Software Factory

Die Web Client Software Factory (WCSF) stellt Hilfsmittel zur Erstellung von Webanwendungen zur Verfügung. Dazu existieren zahlreiche Beispiele und wieder verwendbarer Code zur Verfügung. Die Webanwendung wird konsequent modular aufgebaut (Plug-In). Auch wird ASP.NET AJAX unterstützt.

Die Web Client Software Factory deckt eine umfangreiche Palette vorhandener Funktionalität ab. Es sind dies:

##### Architektur (strukturell)

Presentation Layer  
Resource Access Layer  
Business Layer

##### Modul (funktionell)

Modularity  
Security  
Manageability  
Lifecycle

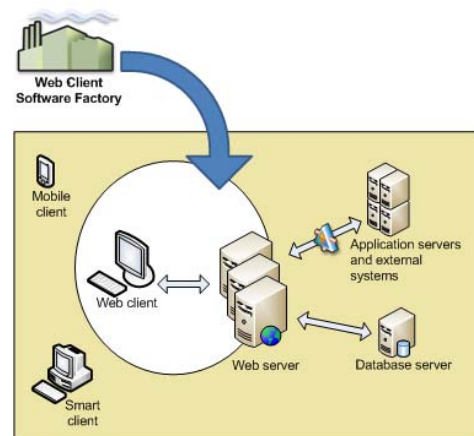


Abbildung 23: Anwendungsbereich der WCSF

Weitere Informationen über die WCSF sind auf der MSDN Seite ausführlich dokumentiert. Die Links sind im Anhang B angegeben.

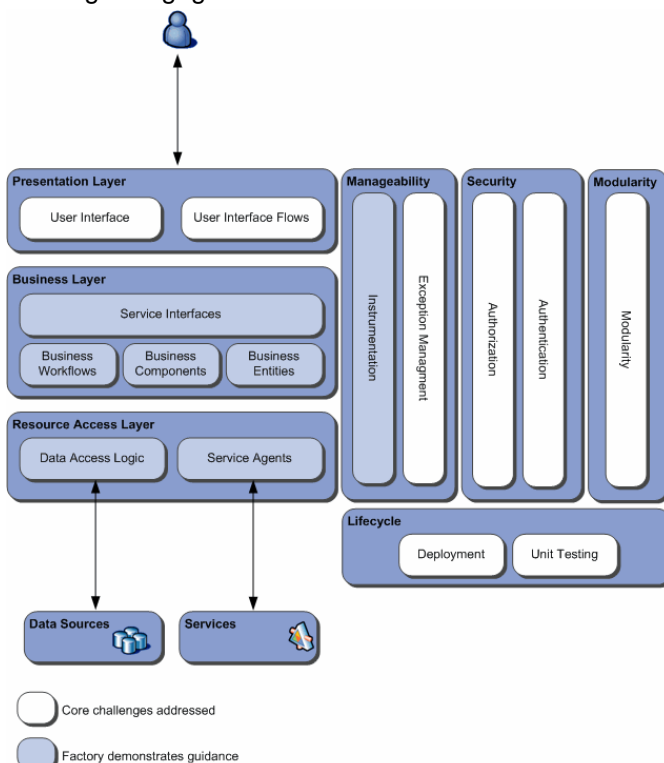


Abbildung 24: Die in der WCSF vorhandene Funktionalität

### 3.1.2 Web Service Software Factory

Die Web Service Software Factory ermöglicht das Modellieren von WCF und ASMX Web Diensten. Dabei werden die benötigten Komponenten zusammengestellt und die Parameter dazu konfiguriert. Anschließend hat man die Möglichkeit, sämtliche Einstellungen zueinander zu validieren. Nach erfolgreicher Validierung kann man direkt alle benötigten Klassen generieren lassen.

Erstellt werden partielle Klassen, welche das Grundgerüst für den Dienst darstellen. Die benötigten Implementationen können nun in weiteren partiellen Klassen sowie Hilfsklassen vorgenommen werden.

Vorteil dieser partiellen Klassen ist, dass bei Änderungen die generierten Klassen neu erstellt werden können und die eigenen Implementationen erhalten bleiben.

Die intern verwendeten Patterns der Service Factory sind auf der MSDN Seite ausführlich dokumentiert. Die Links sind im Anhang B angegeben.

Die Projektstruktur wird durch die WSSF vorgegeben und ist somit für alle Dienste gleich. Am Beispiel des Auftragsservers wird diese aufgezeigt.

Alle für den WCF Dienst benötigte Klassen werden mit Hilfe der WSSF generiert. Die Implementation des Dienstes erfolgte im Projekt der Service Implementation.

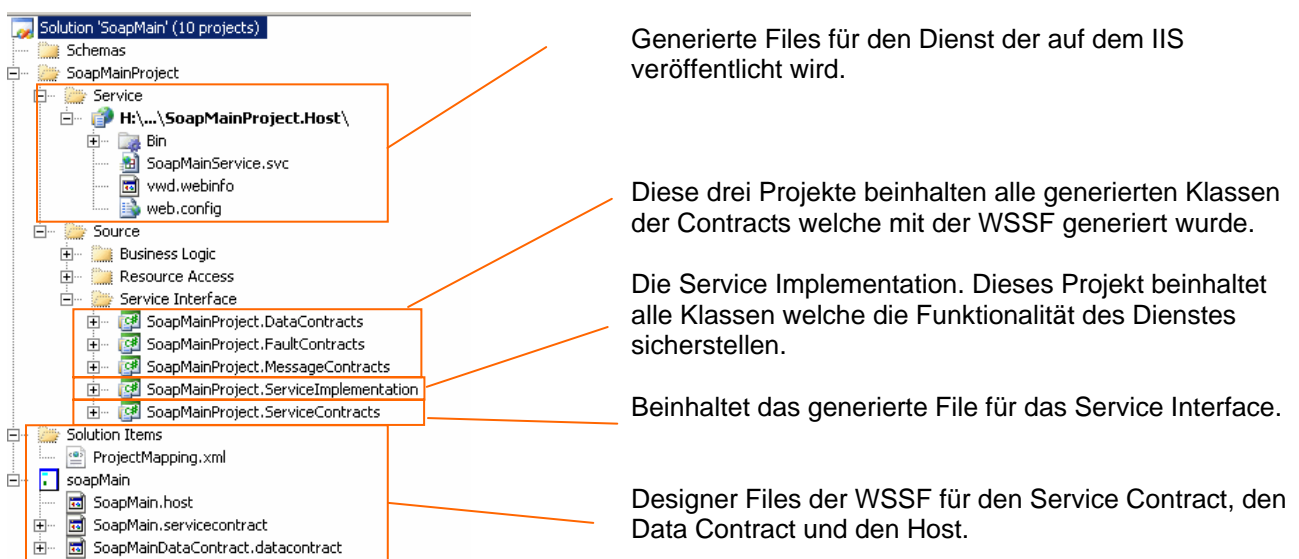


Abbildung 25: Auftragsserver, Struktur der Solution

## 3.2 Systemklassenbeschreibung

Die Systemklassen werden zum grossen Teil bereits mit den Software Factories generiert oder durch Assistenten (z.B. LINQ to SQL Template) erstellt.

Bei den Diensten wird nur bei der Service Implementation Hand angelegt (Implementation des Dienstes, Fassadenpattern für DB Controller, Logger, Berechnungsserver sowie eigene Fehlerklassen). Alle übrigen Klassen werden mit der WSSF generiert.

Die WCSF generiert jeweils die Hülle der Klasse, welche dann auscodiert werden.

Hinzu kommen die zusätzlich benötigten Klassen, welche manuell angelegt werden.

### 3.2.1 Auftragsserver

Der Auftragsserver stellt die Auftragsabwicklung sicher. Dies beinhaltet neben der Logik für die Auftragsabwicklung auch Komponenten für den Datenbankzugriff, einen File Store wo die benötigten Dateien gespeichert werden, die Verwaltung der Berechnungsserver und die Funktionalität für die Weboberfläche. Implementiert ist der Auftragsserver als WCF Dienst.

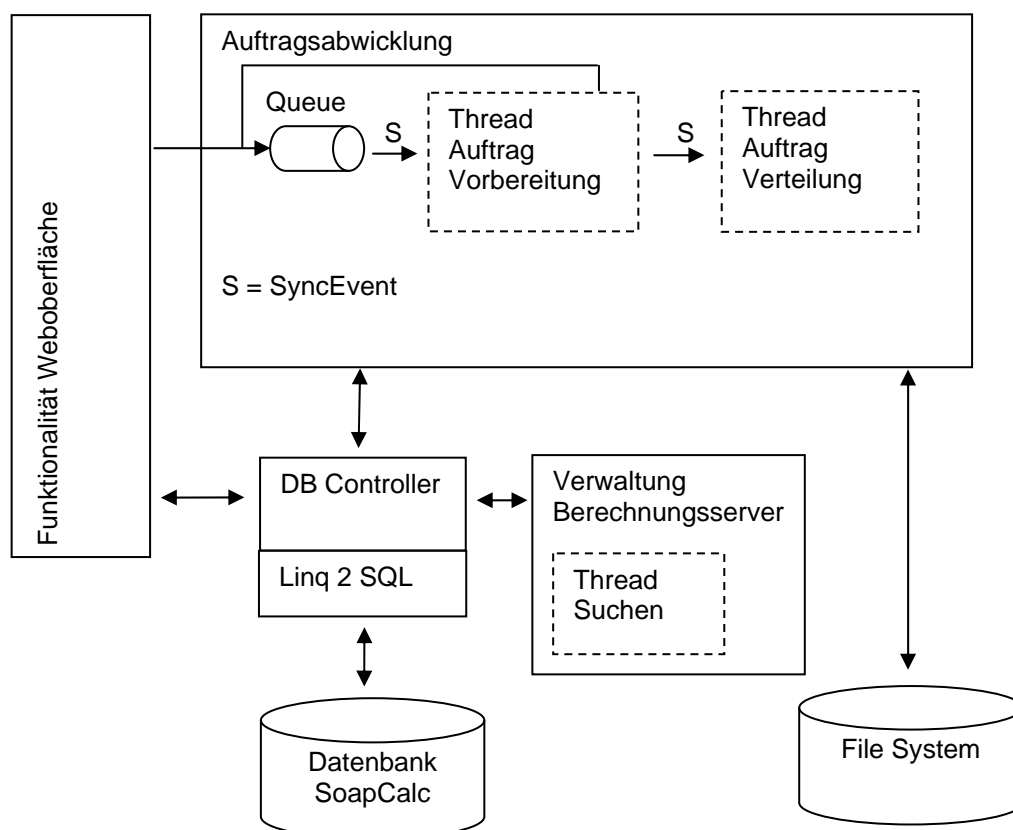


Abbildung 26: Schema Auftragsserver

Für die Auftragsabwicklung werden die Queue und die beiden Threads verwendet.

Sämtliche benötigten Datenbank-Zugriffe werden über den DB Controller abgewickelt, welcher wiederum auf den LINQ2SQL Data Context zugreift.

Die Weboberfläche nutzt und steuert den Auftragsserver über den WCF Dienst. Auch alle Statusinformationen werden über den Dienst abgefragt.



Das Klassendiagramm zeigt die Verbindung der einzelnen Komponenten:

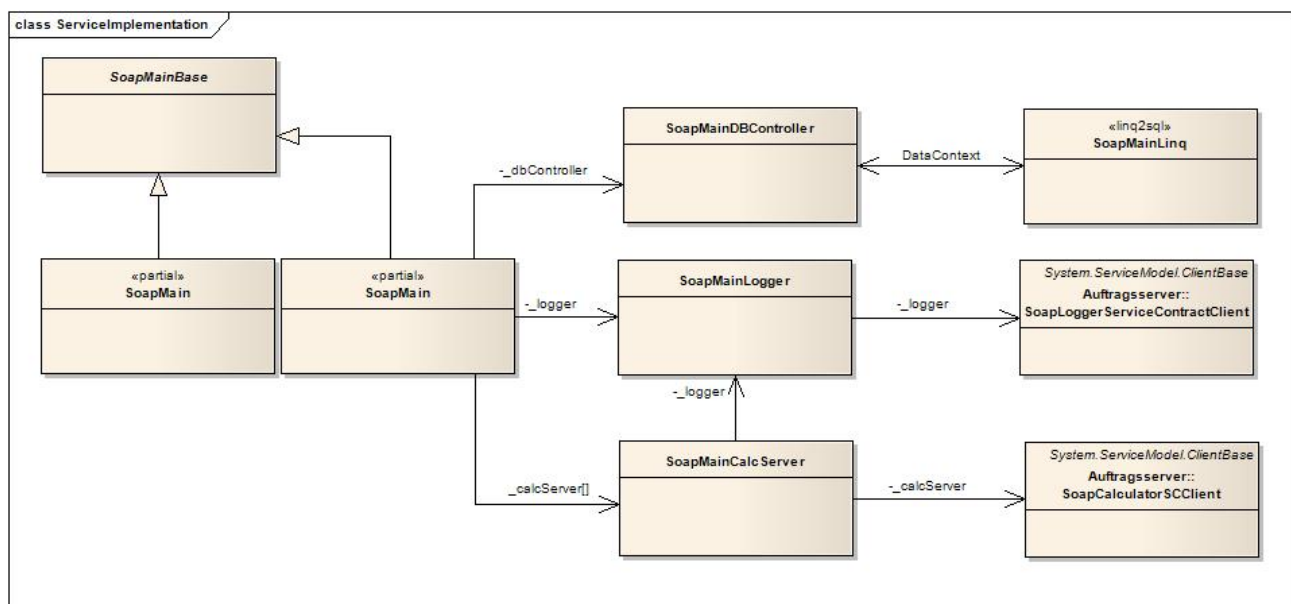


Abbildung 27: Klassendiagramm Auftragsserver

Abgeleitet werden die beiden partiellen Klassen SoapMain von der Klasse SoapMainBase. Die Klasse SoapMainBase sowie die eine partielle Klasse SoapMain werden von der WSSF generiert. Die zweite partielle SoapMain Klasse beinhaltet die Implementierung des Auftragsservers. Für die Implementation benötigt werden je eine Proxyklasse für die WCF Dienste Logger und Berechnungsserver sowie eine Linq2SQL Klasse für die Anbindung der Datenbank. Diese drei Klassen sind jeweils mit einem Fassadenpattern in der Implementation des Auftragsservers eingebunden.

### 3.2.1.1 Service Contract

Der Service Contract beinhaltet alle Remote Methoden des WCF Dienstes sowie die dazugehörigen Message Contracts.

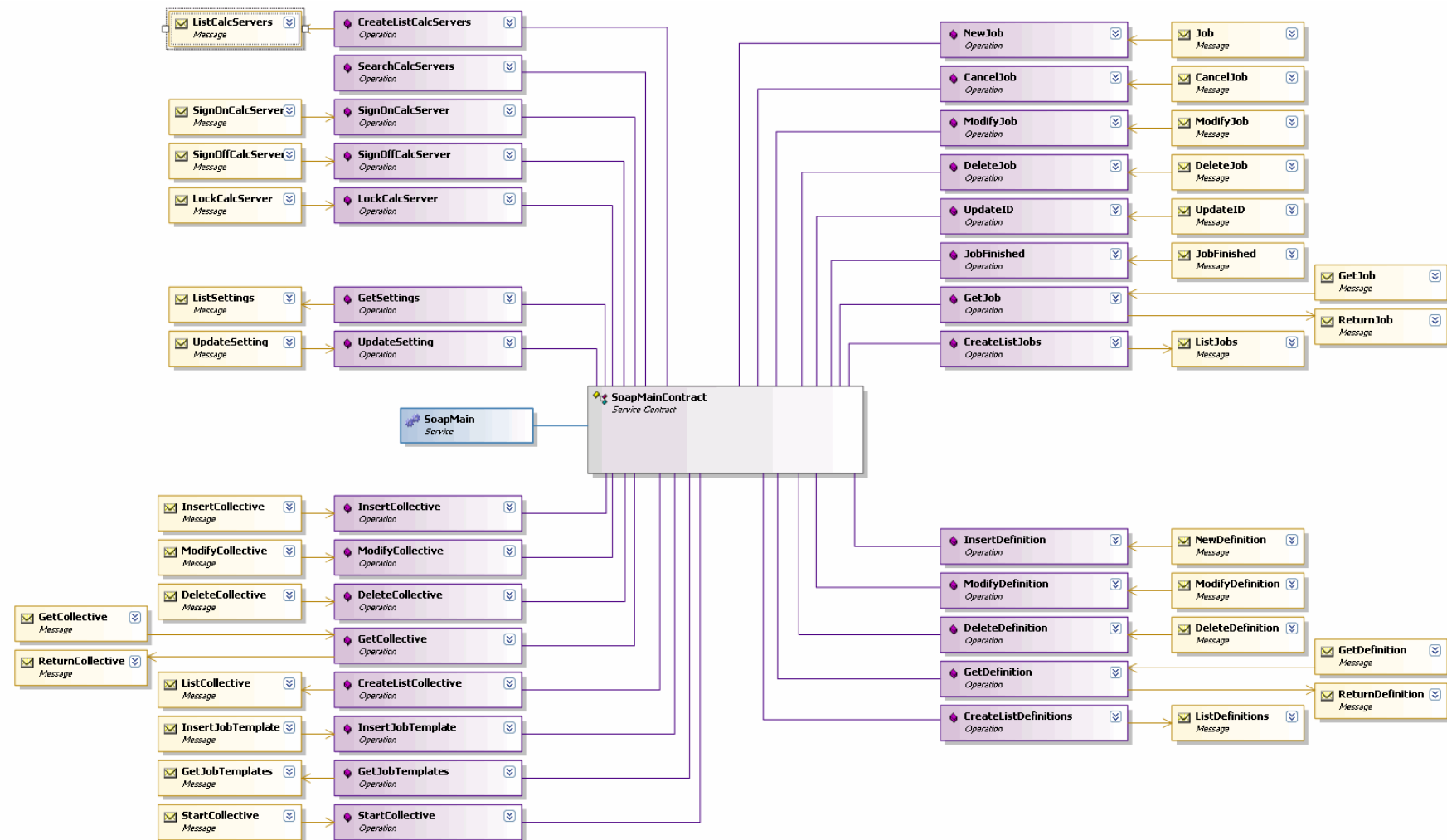


Abbildung 28: SoapMain Service Contract

### 3.2.1.1.1 Service Contract, Teil der Aufträge

Folgende Methoden werden vom Dienst für die Verarbeitung von Aufträgen zur Verfügung gestellt. Bei allen wird in einem Fehlerfall ein JobFault zurückgegeben.

#### *NewJob()*

Diese Methode nimmt einen neuen Auftrag entgegen, schreibt diesen in die Queue und feuert einen SyncEvent. Damit wird die Vorbereitung gestartet.

#### *CancelJob()*

Anhand der übergebenen Guid wird der Auftrag in der DB gesucht. Wenn dieser Status Bereit oder Warten hat, wird der Status auf Abgebrochen gesetzt. Falls der Status In Bearbeitung ist, wird die Methode killProcess() vom Dienst des Berechnungsservers aufgerufen, welche den Auftrag auf dem Berechnungsserver abschiesst.

#### *ModifyJob()*

Der geänderte Auftrag wird entgegen-genommen und an den DB Controller weitergeleitet.

#### *DeleteJob()*

Die Guid des zu löschenden Auftrags wird an den DB Controller weitergeleitet.

#### *UpdateID()*

Der Berechnungsserver kann mit dieser Methode die Process ID sowie den Servernamen des Auftrags mit der mitgegebenen Guid aktualisieren. Diese werden an den DB Controller weitergeleitet.

#### *JobFinished()*

Der Berechnungsserver meldet das Ende der Verarbeitung des Auftrags mit dem Aufruf dieser Methode. Zusätzlich zur Guid des Auftrags wird noch der Exit Code mitgegeben. Bei einer erfolgreichen Beendigung ist dieser 0. Nach Auswertung des Exit Codes wird das Resultat an den DB Controller weitergeleitet und eine eMail Benachrichtigung an den Benutzer versendet. Anschliessend werden allfällige Abhängigkeiten gelöscht, damit nachfolgende Aufträge verarbeitet werden können.

#### *GetJob()*

Diese Methode gibt einen Auftrag zurück. Für den Fall, dass die mitgegebene Guid leer ist (Alle Zahlen 0), wird eine solche generiert, der Ordner im File Store angelegt und ein neues Objekt mit den Angaben für einen neuen Auftrag gesetzt. Anschliessend wird das neue Objekt zurückgegeben. Wenn die Guid nicht leer ist, wird diese an den DB Controller übergeben. Dieser gibt das Objekt mit den Angaben des Auftrags zurück.

#### *CreateListJobs()*

Eine Liste aller Aufträge wird mit dem Aufruf dieser Methode über den DB Controller erstellt und zurückgegeben.

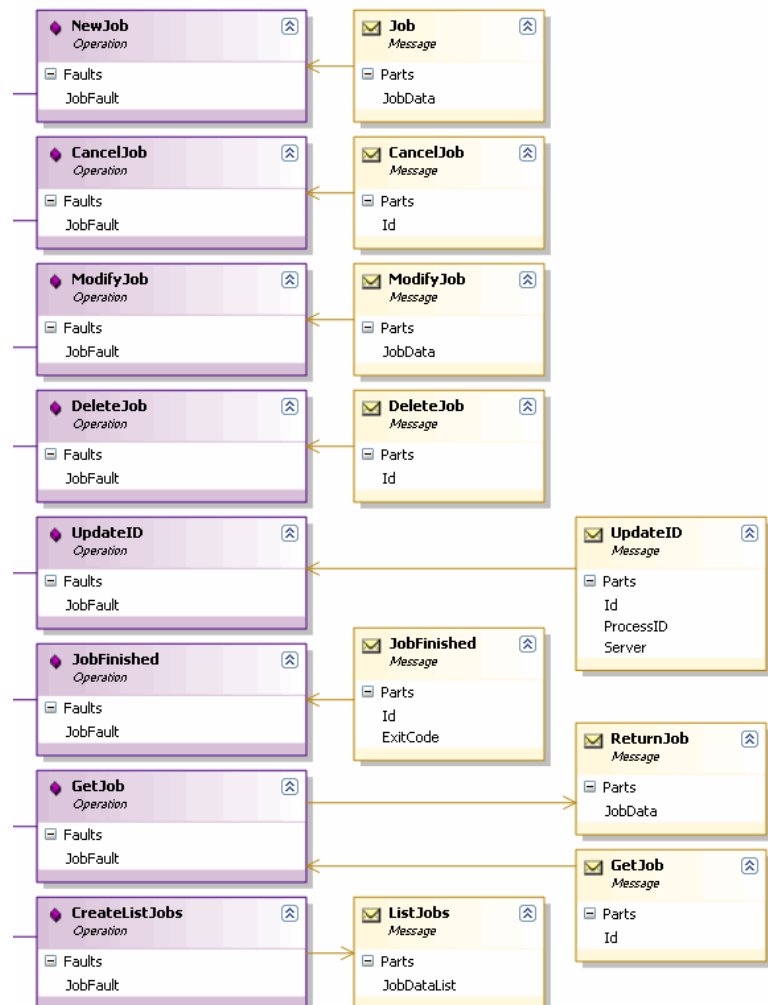


Abbildung 29: SoapMain Service Contract, Teil der Auftragsverarbeitung

### 3.2.1.1.2 Service Contract, Teil der Auftragsdefinitionen

Folgende Methoden werden vom Dienst für die Auftragsdefinitionen zur Verfügung gestellt. Bei allen wird in einem Fehlerfall ein JobFault zurückgegeben.

#### *InsertDefinition()*

Nimmt eine neue Auftragsdefinition entgegen und leitet sie an den DB Controller weiter.

#### *ModifyDefinition()*

Nimmt eine geänderte Auftragsdefinition entgegen und leitet sie an den DB Controller weiter.

#### *DeleteDefinition()*

Die Guid der zu löschenden Auftragsdefinition wird an den DB Controller weitergeleitet.

#### *GetDefinition()*

Diese Methode gibt eine Auftragsdefinition zurück. Für den Fall, dass die mitgegebene Guid leer ist, wird eine solche generiert, der Ordner im File Store angelegt und ein neues Objekt mit den Angaben für eine neue Auftragsdefinition gesetzt.

Anschliessend wird das neue Objekt zurückgegeben. Wenn die Guid nicht leer ist, wird diese an den DB Controller übergeben. Dieser gibt das Objekt mit den Angaben der Auftragsdefinition zurück.

#### *CreateListDefinition()*

Eine Liste aller Auftragsdefinitionen wird mit dem Aufruf dieser Methode über den DB Controller erstellt und zurückgegeben.

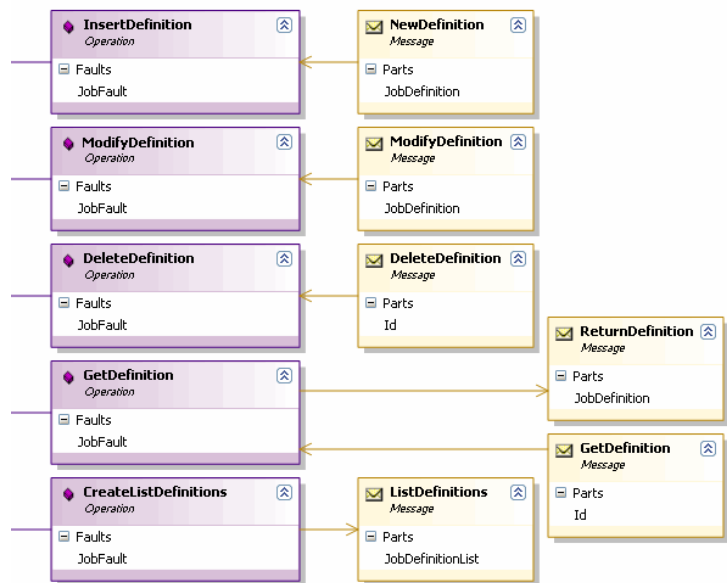


Abbildung 30: SoapMain Service Contract, Teil der Auftragsdefinitionen

### 3.2.1.1.3 Service Contract, Teil der Sammelaufträge

Folgende Methoden werden vom Dienst für die Sammelaufträge zur Verfügung gestellt. Bei allen wird in einem Fehlerfall ein JobFault zurückgegeben.

#### *InsertCollective()*

Nimmt einen neuen Sammelauftrag entgegen und leitet diesen an den DB Controller weiter.

#### *ModifyCollective ()*

Nimmt geänderten Sammelauftrag entgegen und leitet diesen an den DB Controller weiter.

#### *DeleteCollective()*

Die Guid des zu löschenden Sammelauftrags wird an den DB Controller weitergeleitet.

#### *GetCollective()*

Diese Methode gibt einen Sammelauftrag zurück. Für den Fall das die mitgegebene Guid leer ist, wird ein solcher generiert und das neue Objekt zurückgeben. Wenn die Guid nicht leer ist, wird diese an den DB Controller übergeben. Dieser gibt das Objekt mit den Angaben des Sammelauftrags zurück.

#### *CreateListCollective()*

Eine Liste aller Sammelaufträge wird mit dem Aufruf dieser Methode über den DB Controller erstellt und zurückgegeben.

#### *InsertJobTemplate()*

Eine neue Auftragsvorlage wird entgegengenommen und an den DB Controller weitergeleitet.

#### *GetJobTemplates()*

Eine Liste aller Auftragsvorlagen wird mit dem Aufruf dieser Methode über den DB Controller erstellt und zurückgegeben.

#### *StartCollective()*

Mit der mitgegebenen Guid wird der entsprechende Sammelauftrag über den DB Controller abgeholt. Anschliessend wird aus allen Auftragsvorlagen ein neuer Auftrag generiert und vorbereitet. Am Schluss werden die Abhängigkeiten gesetzt. Ein SyncEvent startet anschliessend die übliche Verteilung der neu erstellen Aufträge

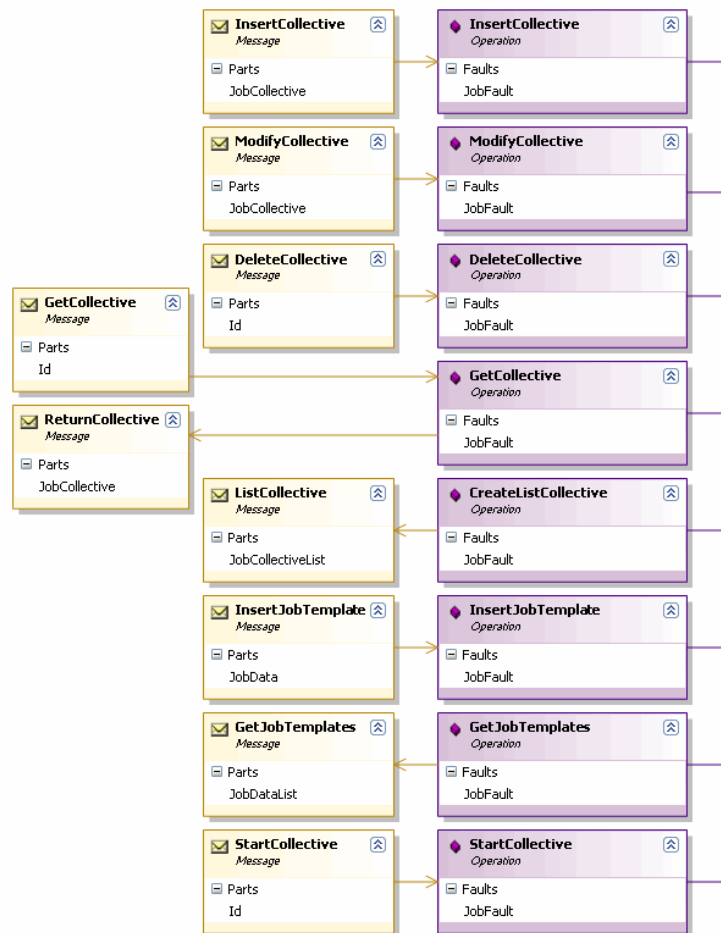


Abbildung 31: SoapMain Service Contract, Teil der Sammelaufträge

### 3.2.1.1.4 Service Contract, Teil der Berechnungsserver

Folgende Methoden werden vom Dienst für die Verwaltung der Berechnungsserver zur Verfügung gestellt. Bei allen wird in einem Fehlerfall ein CalcServer Fault zurückgegeben.

#### *CreateListCalcServers()*

Eine Liste aller angemeldeten Berechnungsserver wird erstellt und zurückgegeben.

#### *SearchCalcServers()*

Ein neuer Thread wird gestartet welcher neue Berechnungsserver sucht.

#### *SignOnCalcServer()*

Diese Methode wird zurzeit nicht mehr verwendet, bleibt aber für allfällige Erweiterungen bestehen.

#### *SignOffCalcServer()*

Diese Methode wird vom Berechnungsserver beim Beenden des Dienstes aufgerufen, damit sich dieser abmelden kann. Die Angaben werden an den DB Controller weitergeleitet.

#### *LockCalcServer()*

Um einen Berechnungsserver zu sperren, beziehungsweise zu entsperren, steht diese Methode zur Verfügung. Dabei wird das Flag an den DB Controller weitergegeben.

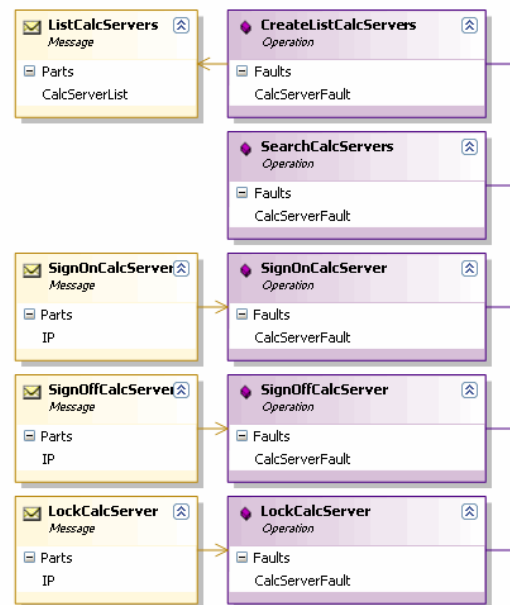


Abbildung 32: SoapMain Service Contract, Teil der Berechnungsserver

### 3.2.1.1.5 Service Contract, Teil der Einstellungen

Folgende Methoden werden vom Dienst für die Ansicht der Einstellungen zur Verfügung gestellt. Bei beiden wird in einem Fehlerfall ein SoapSettingsFault zurückgegeben.

#### *GetSettings()*

Eine Liste aller Settings wird erstellt und zurückgegeben.

#### *UpdateSetting()*

Ermöglicht es, ein Setting neu zu definieren. Da aber Application- und nicht User Settings eingesetzt werden müssen, wird diese Methode nicht verwendet.

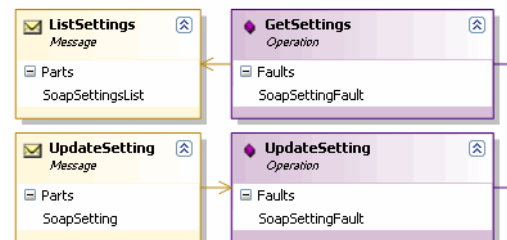


Abbildung 33: SoapMain Service Contract, Teil der Einstellungen

### 3.2.1.2 Data und Fault Contract

Im Data Contract sind die Business Objects abgebildet. Die daraus erstellten Objekte werden für den WCF Dienst verwendet und können so übertragen werden. Erstellt wurde jeweils pro Business Object je eine Klasse sowie eine Collection, welche eine Liste des jeweiligen Typs enthält.

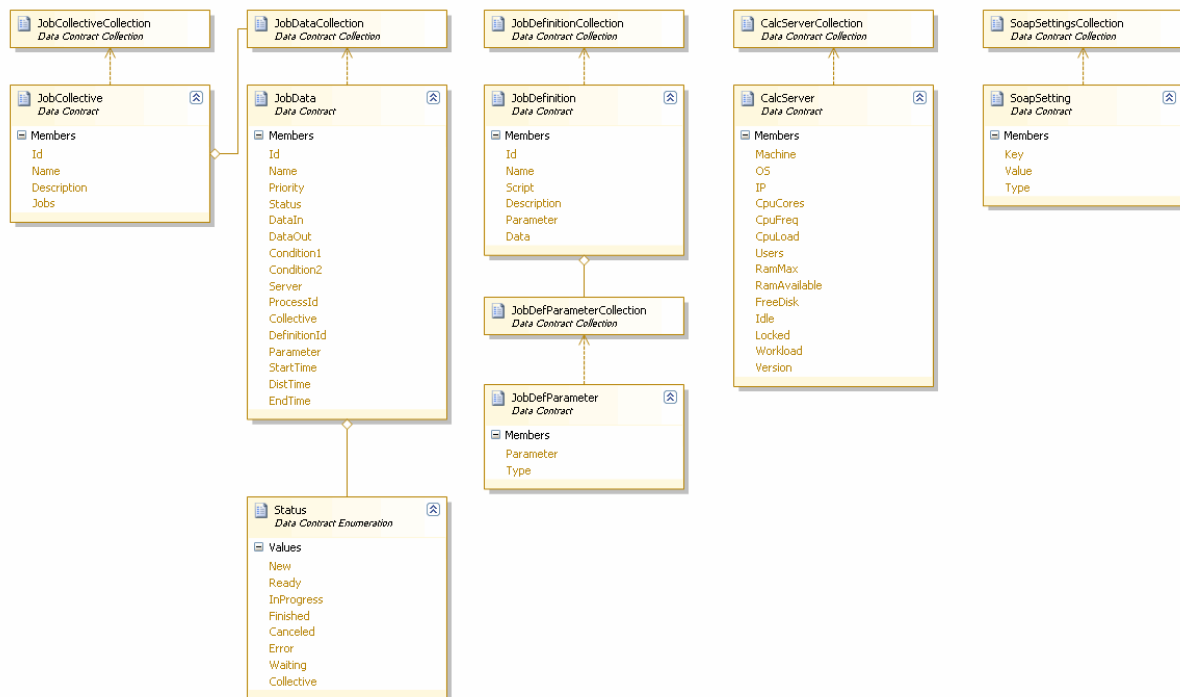


Abbildung 34: SoapMain Data Contract

Die Fault Contracts sind spezifische Klassen für das Exceptionhandling welche von der Klasse Fault Exception abgeleitet wurden. Zur Laufzeit werden die Fault Exceptions von der WCF in SOAP Exceptions umgewandelt. Diese Soap Exceptions werden über den Dienst versendet. Für alle Bereiche wurde eine eigene Klasse erstellt. Alle beinhalten ein Feld Description für die genauere Beschreibung des Fehlers.

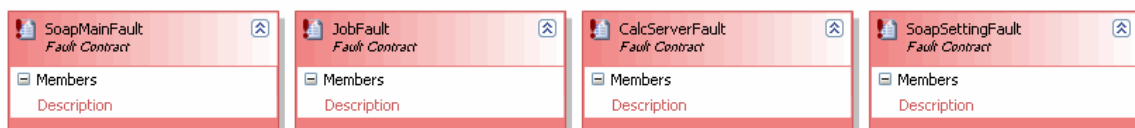


Abbildung 35: SoapMain Fault Contracts

### 3.2.1.3 Host Model

Das Host Model definiert die Dienst-Endpunkte. Als Binding Type wird basicHttpBinding verwendet.

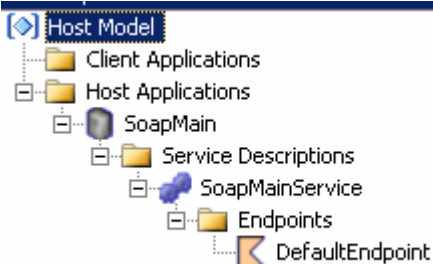


Abbildung 36: SoapMain Host Model

### 3.2.1.4 DB Controller und Datenbanktabellen

Sämtliche Datenbankzugriffe werden durch die Klasse SoapMainDBController gesteuert. Für den Zugriff werden Data Contexte von LINQ2SQL eingesetzt. Die Namen der Methoden sind jeweils selbstsprechend und beschreiben was auf der Datenbank ausgeführt wird.

Zusätzlich zu den DB Zugriffen werden noch einige Cast Methoden verwendet um die Objekte vom WCF Dienst in die von LINQ2SQL umzupacken, bzw. umgekehrt. Diese enthalten dieselben Properties, sind aber miteinander nicht kompatibel.

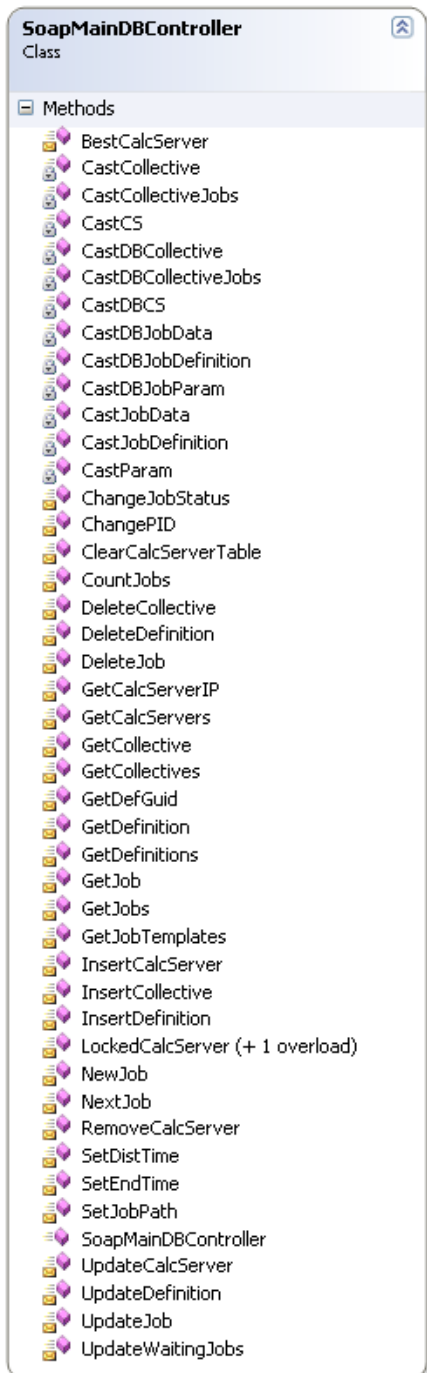


Abbildung 37: Die Klasse SoapMainDBController



Die Business Objects (siehe auch Kap. 2.5.3) werden in verschiedenen Tabellen gespeichert. Die Definitionen dieser Tabellen sehen wie folgt aus:


 id	uniqueidentifier	<input type="checkbox"/>
name	varchar(50)	<input checked="" type="checkbox"/>
priority	int	<input checked="" type="checkbox"/>
status	int	<input checked="" type="checkbox"/>
data_in	varchar(MAX)	<input checked="" type="checkbox"/>
data_out	varchar(MAX)	<input checked="" type="checkbox"/>
definition	uniqueidentifier	<input checked="" type="checkbox"/>
condition1	uniqueidentifier	<input checked="" type="checkbox"/>
condition2	uniqueidentifier	<input checked="" type="checkbox"/>
server	varchar(50)	<input checked="" type="checkbox"/>
process_id	int	<input checked="" type="checkbox"/>
parameter	varchar(MAX)	<input checked="" type="checkbox"/>
start_time	datetime	<input type="checkbox"/>
dist_time	datetime	<input type="checkbox"/>
end_time	datetime	<input type="checkbox"/>
collective	uniqueidentifier	<input checked="" type="checkbox"/>

Abbildung 38: Tabelle JobsDB


 machine	varchar(50)	<input type="checkbox"/>
os	varchar(50)	<input checked="" type="checkbox"/>
ip	varchar(15)	<input checked="" type="checkbox"/>
users	int	<input checked="" type="checkbox"/>
cpu_cores	int	<input checked="" type="checkbox"/>
cpu_freq	int	<input checked="" type="checkbox"/>
cpu_load	int	<input checked="" type="checkbox"/>
ram_max	bigint	<input checked="" type="checkbox"/>
ram_available	bigint	<input checked="" type="checkbox"/>
free_disk	bigint	<input checked="" type="checkbox"/>
idle	bit	<input checked="" type="checkbox"/>
locked	bit	<input checked="" type="checkbox"/>
workload	float	<input checked="" type="checkbox"/>

Abbildung 39: Tabelle CalcServerDB


 id	uniqueidentifier	<input type="checkbox"/>
name	varchar(50)	<input checked="" type="checkbox"/>
script	varchar(50)	<input checked="" type="checkbox"/>
description	varchar(MAX)	<input checked="" type="checkbox"/>

Abbildung 40: Tabelle DefinitionsDB

 auto	int	<input type="checkbox"/>
id	uniqueidentifier	<input type="checkbox"/>
parameter	varchar(50)	<input type="checkbox"/>
type	varchar(50)	<input checked="" type="checkbox"/>

Abbildung 41: Tabelle DefinitionParamsDB


 id	uniqueidentifier	<input type="checkbox"/>
name	varchar(50)	<input checked="" type="checkbox"/>
description	varchar(MAX)	<input checked="" type="checkbox"/>

Abbildung 42: Tabelle CollectiveDB


 auto	int	<input type="checkbox"/>
collectiv_id	uniqueidentifier	<input checked="" type="checkbox"/>
job_id	uniqueidentifier	<input checked="" type="checkbox"/>
condition1	uniqueidentifier	<input checked="" type="checkbox"/>
condition2	uniqueidentifier	<input checked="" type="checkbox"/>
name	varchar(50)	<input checked="" type="checkbox"/>

Abbildung 43: Tabelle CollectiveJobsDB

Nicht ersichtlich sind hier die Foreign-Keys Beziehungen zwischen der Tabelle DefinitionsDB (id) und DefinitionParamsDB (id), JobsDB(definition) und DefinitionsDB (id) sowie zwischen der Tabelle CollectiveDB (id) und CollectiveJobsDB (collectiv\_id).

Die beiden „auto Spalten“ werden nur benötigt, weil LINQ2SQL falsche Resultate lieferte, wenn ein Primärschlüssel über mehrere Spalten definiert wurde.

### 3.2.1.5 Berechnungsserver

Für die Zugriffe auf die Berechnungsserver wird die Klasse SoapMainCalcServer eingesetzt. Diese ist als Fassadenpattern implementiert und kapselt daher alle Aufrufe auf den WCF Dienst bzw. auf die Proxy Klasse des jeweiligen Berechnungsservers.

Auf dem Auftragsserver wird jeweils beim Start auch ein Berechnungsserver initialisiert. Dies stellt sicher, dass immer Berechnungen in einer Minimalumgebung durchgeführt werden können. Zudem werden weitere Server gesucht.

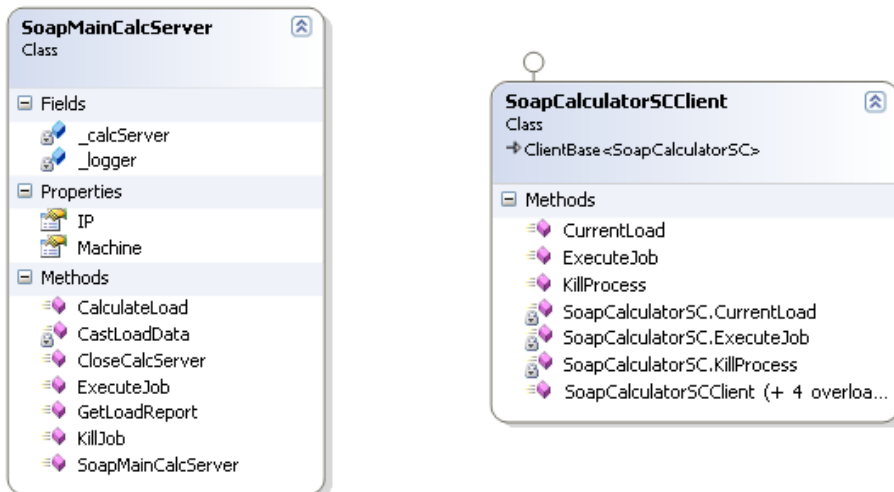


Abbildung 44: Die Klasse SoapMainCalcServer und die Proxyklasse

Zur Suche nicht angemeldeter Berechnungsserver wird der IP Range des Auftragsservers gescannt. Dabei wird der Range des letzten Oktets (0-255) berücksichtigt. Dazu wird in einem eigenen Thread zuerst ein Ping an jede nicht bereits angemeldete IP gesendet und bei einer positiven Antwort versucht den Dienst zu starten. Bei Erfolg wird der neue Berechnungsserver in die DB eingetragen und fortan eingesetzt.

### 3.2.1.6 Logger

Für den Zugriff auf den Logger wird die Klasse SoapMainLogger eingesetzt. Diese ist als Fassadenpattern implementiert und baut die Verbindung zum Dienst über die Proxy Klasse auf und kapselt alle Aufrufe auf den WCF Dienst des zentralen Loggers.

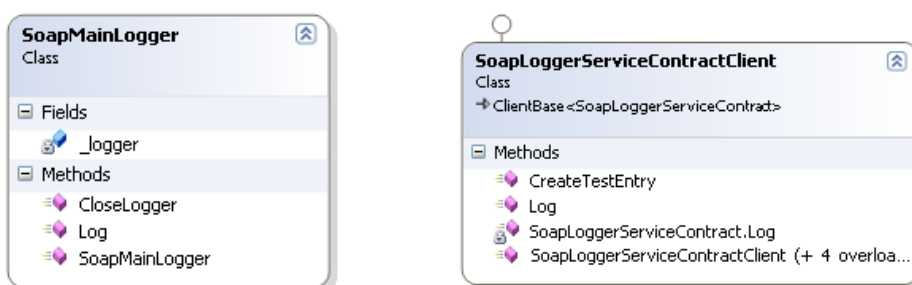


Abbildung 45: Die Klasse SoapMainLogger und die Proxyklasse

### 3.2.1.7 Auftragsabwicklung

Für die Auftragsabwicklung werden die Queue und die beiden Threads verwendet. Diese bereiten die Aufträge vor (siehe auch Kap. 2.8.1.2) und verteilen sie anschliessend an die Berechnungsserver (siehe auch Kap. 2.8.1.3). Die Signalisation zwischen den Threads ist durch SyncEvents gesteuert.

Zum komprimieren der Files wird eine OpenSource Library von Ionic mit dem zip Algorithmus eingesetzt.

### 3.2.1.8 SyncEvents

Die SyncEvents werden zur Synchronisation der beiden Threads der Abwicklung verwendet. Dies funktioniert wie bei einem Producer - Consumer Ansatz.

Der Producer feuert einen SyncEvent, nachdem er seinen Ablauf abgeschlossen hat. Hier sind dies die Queue, in welche neue Aufträge abgelegt werden, sowie der Thread, der einen Auftrag vorbereitet hat.

Nach Erhalt des SyncEvents durchläuft der entsprechende Thread jeweils seinen Ablauf. Anschliessend wartet er auf den nächsten Event, ohne zu pollen und dabei Ressourcen zu verschwenden.

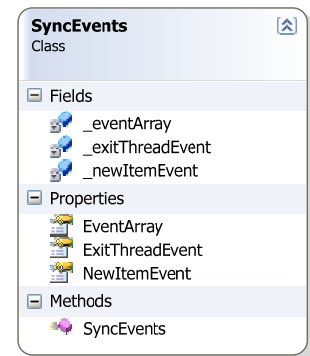


Abbildung 46: Die Klasse SyncEvents

### 3.2.1.9 Einstellungen

Die Einstellungen für den Auftragsserver werden in der Datei web.config vorgenommen. Diese sind in der Sektion AppSettings abgelegt.

```
<appSettings>
  <add key="TimerIntervalMin" value="1" />
  <add key="JobFolder" value="C:\SoapCalc\Jobs" />
  <add key="DefFolder" value="C:\SoapCalc\Definitions" />
  <add key="CollectiveFolder" value="C:\SoapCalc\Collective" />
  <add key="Share" value="\\MB002\SoapCalc\Jobs\" />
  <add key="Notification" value="michael.berger@embe.ch" />
  <add key="MailFrom" value="soapcalc@embe.ch" />
  <add key="minCalcServerVersion" value="1.0" />
</appSettings>
```

Abbildung 47: Die SoapMain Settings

Die Einstellungen für die WCF sind ebenfalls in der Datei web.config abgelegt. Dafür steht die Sektion

```
<system.serviceModel>
```

zur Verfügung. Hier werden die Endpoints, das Verhalten und das Binding konfiguriert.

### 3.2.1.10 Exception Handling

Für die Fehlerbehandlung wurden zwei Exception Klassen angelegt, welche von der Klasse Exception erben. Eine SoapCalcException wird geworfen, falls bei einem Berechnungsserver ein Fehler auftritt. Die SoapDBException wird bei einem Fehlerfall im DB Controller eingesetzt.

### 3.2.1.11 Ausblick

Für den Auftragsserver könnte man folgende Erweiterung vorsehen:

Die Auswahl der Auftragsserver soll frei vom Benutzer definiert oder automatisch vorgenommen werden können. Ausserdem sollte der Filetransfer zwischen dem Auftragsserver und den Berechnungsservern nicht mehr über eine Netzwerkfreigabe, sondern über ein anderes Protokoll wie z.B. FTP abgewickelt werden. Der Vorteil dieser Änderung wäre, dass die Server nicht mehr im gleichen Netzwerk sein müssten. Im Augenblick ist dies so nur mit der Weboberfläche möglich.

Die Funktionalität der Parameter deckt die Anforderungen des Systems. Hier wäre aber noch Verbesserungspotential vorhanden. Als Beispiel könnte man die Parameter bis auf dem Berechnungsserver getrennt hält, sowie die Möglichkeit zum Aufruf der Parameter erweitern. So zum Beispiel ob der Parametername auch angegeben oder ob nur der Wert selbst mitgegeben werden soll.

### 3.2.2 Berechnungsserver

Der Berechnungsserver nimmt die Aufträge entgegen und verarbeitet diese. Dazu wird pro Aufruf ein Thread gestartet der die Kontrolle über den Ablauf übernimmt. Nachdem der Auftrag für die Verarbeitung vorbereitet ist, wird ein neuer Prozess ausserhalb des Prozessraums des IIS gestartet der den Auftrag ausführt. Nach dem Abschluss des Prozesses wird das Resultat an den Auftragsserver zurückgesendet und der Thread beendet.

Weiter stellt der als WCF Dienst implementierte Berechnungsserver Methoden für die Erstellung eines Auslastungsreports sowie zum Abschliessen eines laufenden Prozesses zur Verfügung.

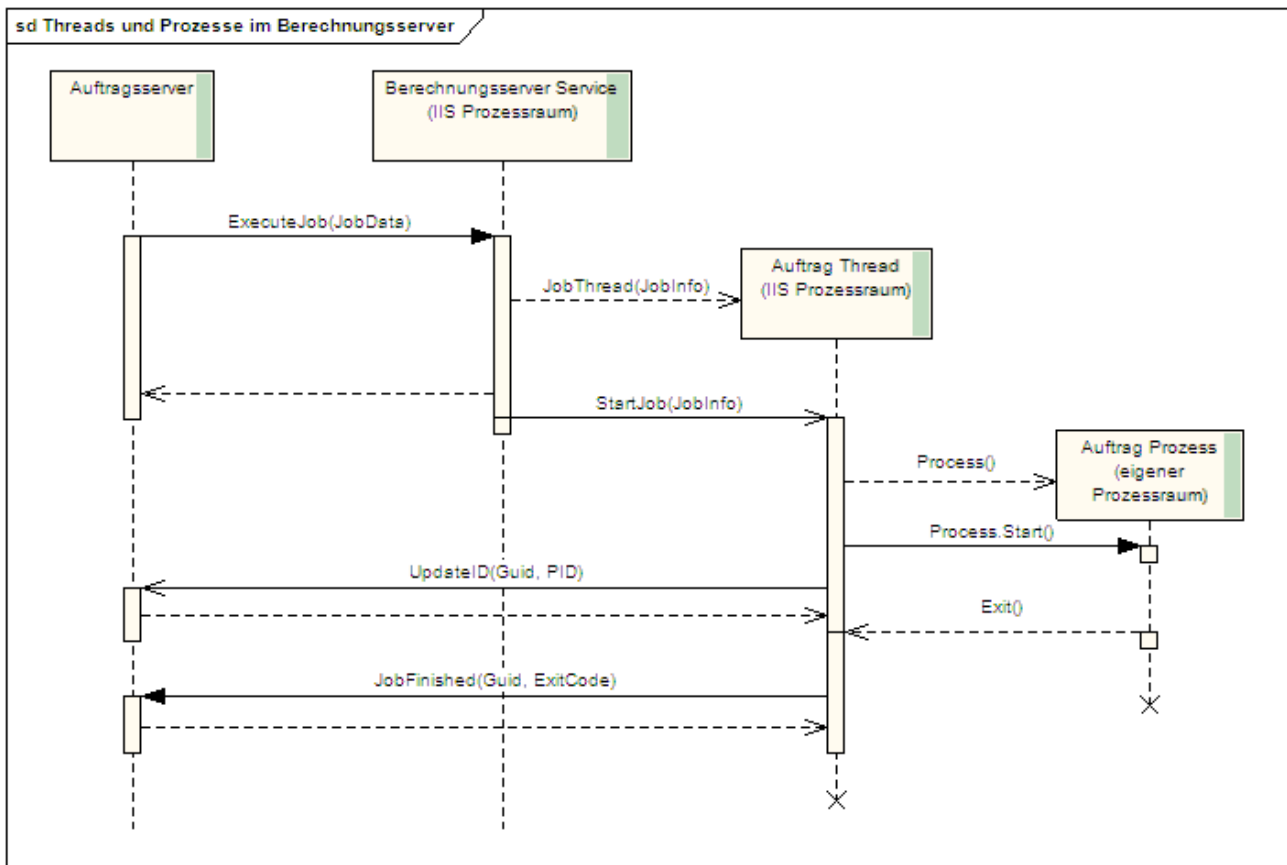


Abbildung 48: Threads und Prozesse des Berechnungsservers

### 3.2.2.1 Service Contract

Folgende Methoden werden vom Dienst für die Verarbeitung von Aufträgen zur Verfügung gestellt. Bei allen wird in einem Fehlerfall eine SoapCalcEx zurückgegeben.

#### *ExecuteJob()*

Diese Methode bildet den Hauptteil des Berechnungsservers. Sie führt die oben beschriebene Auftragsabwicklung durch.

#### *CurrentLoad()*

Die aktuelle Auslastung des Rechners wird ausgelesen. Ermittelt werden die im Business Object definierten Parameter. Diese werden mit Methoden die .NET zur Verfügung stellt, und auch mit WMI-Abfragen (Windows Management Instrumentation), ausgelesen. Zurückgegeben wird ein aktueller Auslastungsreport (siehe auch Kap. 2.5.3.6)

#### *KillProcess()*

Diese Methode beendet einen Prozess in Bearbeitung. Dabei wird versucht, den Prozess mit der mitgegebenen Prozess ID zu übernehmen und abzuschliessen.

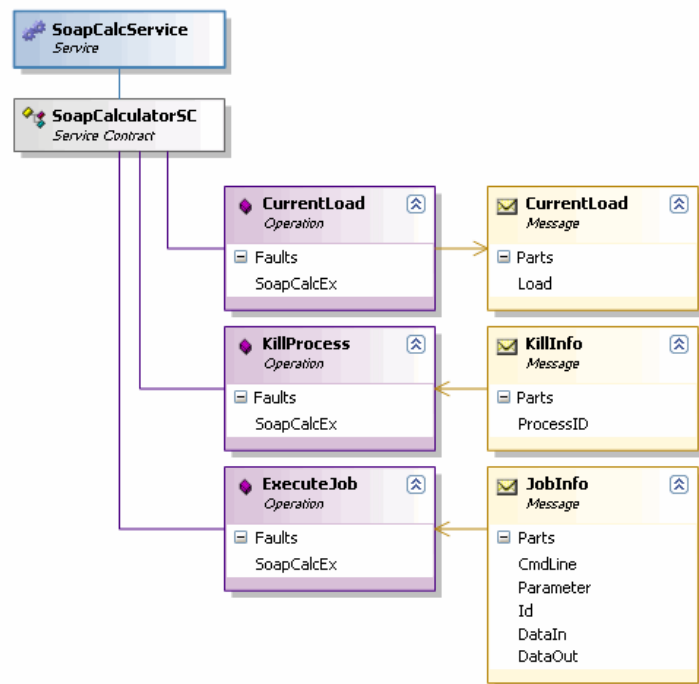


Abbildung 49: SoapCalc Service Contract

### 3.2.2.2 Data- und Fault Contract

Im Data Contract ist das Business Object des Auslastungsreports abgebildet. Die daraus erstellten Objekte werden für den WCF Dienst verwendet und können so übertragen werden.

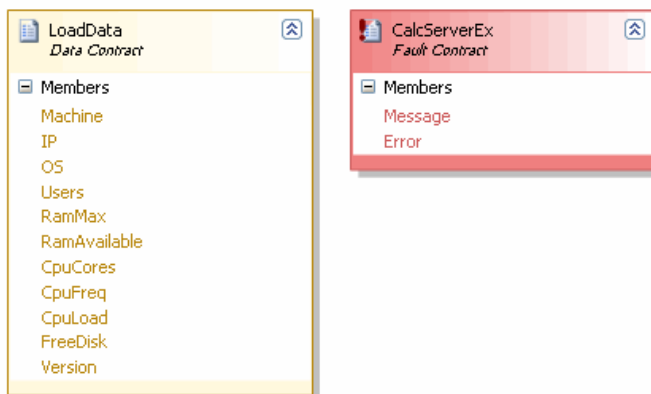


Abbildung 50: SoapCalc Data- und Fault Contract

In einem Fehlerfall zur Laufzeit geworfene Fault Exceptions werden von der WCF in SOAP Exceptions umgewandelt. Diese Soap Exceptions werden über den Dienst versendet.

### 3.2.2.3 Logger

Für den Zugriff auf den Logger wird die Klasse SoapCalcLogger eingesetzt. Diese ist als Fassadenpattern implementiert und baut die Verbindung zum Dienst über die Proxy Klasse auf und kapselt alle Aufrufe auf den WCF Dienst des zentralen Loggers.

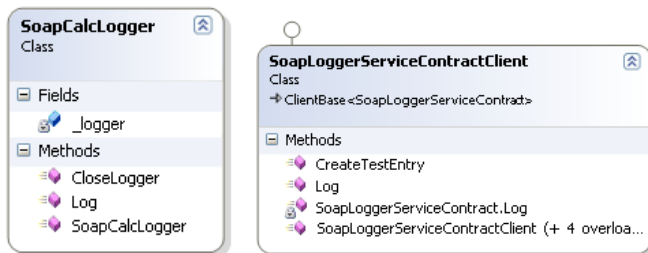


Abbildung 51: Die Klasse SoapCalcLogger und die Proxyklasse

### 3.2.2.4 Einstellungen

Die Einstellungen für die WCF sind in der Datei web.config abgelegt. Dafür steht die Sektion

```
<system.serviceModel>
```

zur Verfügung. Hier werden die Endpunkte, das Verhalten und das Binding konfiguriert.

### 3.2.2.5 Ausblick

Für die Version 2.0 könnte man die Implementation wie folgt erweitern:

Für den Fall, das der IIS aus irgend einem Grund neu gestartet wird, könnte man beim Auftragsserver eine aktuelle Liste der Aufträge die auf diesem Berechnungsserver laufen abholen und versuchen, pro Prozess einen Thread zu generieren und so wieder die Kontrolle über den Prozess bekommen.

Für den Fall das ein benötigtes Softwarepaket nicht installiert ist, kann man die Auftragsabwicklung so erweitern, dass ein Auftrag abgelehnt werden kann.

### 3.2.3 Log Server

Ein zentraler WCF Dienst SoapLogger steht allen Komponenten von SoapCalc zur Verfügung. Er nimmt einen Log Eintrag entgegen und schreibt diesen anschliessend in die Datenbank.

Beim ersten Aufruf des Dienstes wird ein separater statischer Thread gestartet, welcher die Verbindung mit der Datenbank aufbaut. Der Logger ist nun bereit.

Ein neuer Log Eintrag wird entgegen genommen. Dazu wird er in eine Queue geschrieben und ein SyncEvent gefeuert. Dieser dient der Synchronisation zwischen dem Thread des Dienstes und des DB Threads.

Durch den SyncEvent angestossen wird nun der DB Thread aktiv. Der Log Eintrag wird wieder aus der Queue ausgelesen und über einen Data Context mit LINQ in die Datenbank eingetragen. Der DB Thread ist solange aktiv, wie Log Einträge in der Queue vorhanden sind. Anschliessend wartet der Thread auf den nächsten Event.

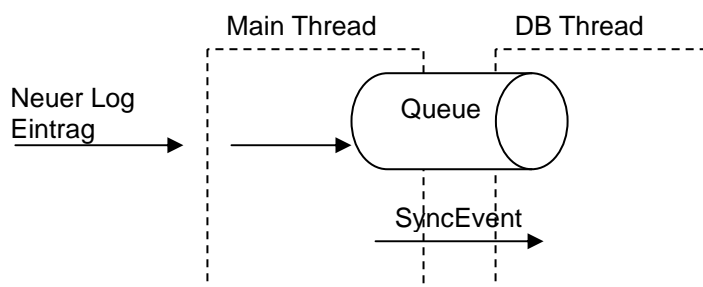


Abbildung 52: Schema Log Server

Eine Queue wird eingesetzt um die beiden Threads zu trennen. Zudem können so Log Einträge rasch an den Log Server abgesetzt werden, ohne dass die Einträge in der DB gespeichert werden müssen. Die Abarbeitung der Queue wird direkt gestartet, die benötigte Zeit ist aber durch die Trennung sekundär.

Der SyncEvent dient dazu, dass der DB Thread nur läuft falls effektiv Daten in die DB eingetragen werden müssen.

#### 3.2.3.1 Service Contract

Der Dienst stellt zwei Methoden zur Verfügung. Von beiden wird im Fehlerfall eine Soap Exception geworfen.

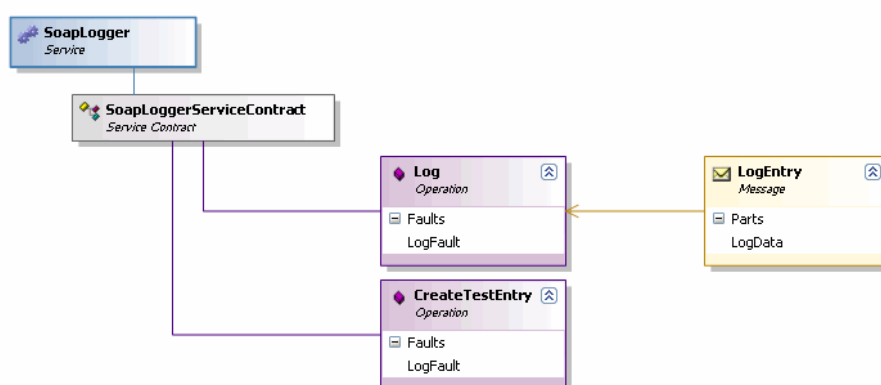


Abbildung 53: SoapLogger Service Contract

#### Log()

Die Methode log steht dem Client zum loggen zur Verfügung. Übergeben wird jeweils ein LogData Objekt, welches die Informationen zum Log Eintrag enthält.

#### CreateTestEntry()

Die Methode CreateTestEntry wird ohne Parameter aufgerufen und generiert einen Test Eintrag in der Datenbank.

### 3.2.3.2 Data und Fault Contract

Für den Log Server ist der Data Contract LogData das zentrale Element. Dieser setzt sich aus den Members Time, Machine, Source und Data zusammen. Der LogLevel wurde als Enumeration realisiert. Die LogDataCollection ermöglicht es, eine Liste von LogDatas zu übergeben. Diese wird beim Log Analyzer eingesetzt.

Für die Fehlerbehandlung wurden zwei Fault Contracts definiert, je eine für den Logger und für den Log Analyzer. Beide beinhalten ein Feld Description für die Beschreibung des Fehlers. In einem Fehlerfall zur Laufzeit geworfene Fault Exceptions werden von der WCF in SOAP Exceptions umgewandelt. Diese Soap Exceptions werden über den Dienst versendet.

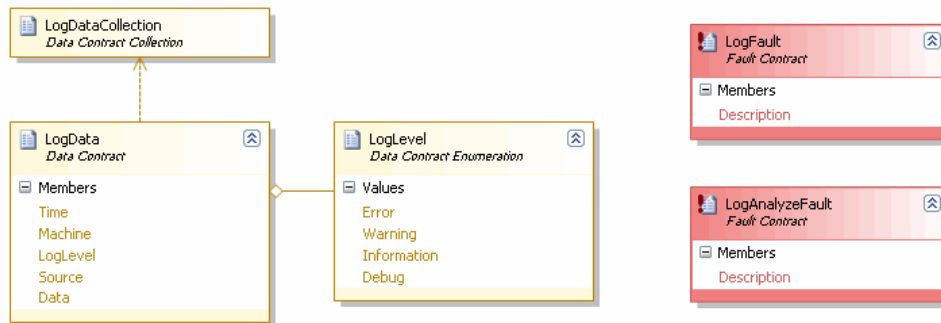


Abbildung 54: SoapLogger Data- und Fault Contract

### 3.2.3.3 Datenbanktabelle

Der Log Server speichert seine Daten in die Tabelle LoggerDB.

	Spaltenname	Datentyp	NULL zula...
	id	int	<input type="checkbox"/>
	timestamp	datetime	<input type="checkbox"/>
	timestamp_db	datetime	<input checked="" type="checkbox"/>
	machine	varchar(50)	<input checked="" type="checkbox"/>
	loglevel	int	<input checked="" type="checkbox"/>
	source	varchar(50)	<input checked="" type="checkbox"/>
	data	varchar(MAX)	<input type="checkbox"/>

Abbildung 55: Definition Tabelle LoggerDB

Das Feld id ist ein Autowert und wird als Primärschlüssel verwendet. Im Feld timestamp\_db wird jeweils bei einem Insert die aktuelle Zeit eingetragen. So können allfällige Schreibverzögerungen erkannt werden. Die übrigen Felder korrespondieren mit dem Inhalt des Data Contracts LogData.

Auf die DB wird mit einem LINQ Data Context zugegriffen. Da LINQ und die Data Contracts beide eigene Objekte zur Verfügung stellen, müssen diese jeweils umgepackt werden.

### 3.2.3.4 Einstellungen

Der Pfad des LogFiles für den Logger wird in der Datei web.config angegeben. Dieser ist in der Sektion AppSettings abgelegt.

```
<appSettings>
  <add key="txtLogFile" value="C:\Temp\logServer.log"/>
</appSettings>
```

Die Einstellungen für die WCF sind ebenfalls in der Datei web.config abgelegt. Dafür steht die Sektion

```
<system.serviceModel>
```

zur Verfügung. Hier werden die Endpunkte, das Verhalten und das Binding konfiguriert.

### 3.2.3.5 Ausblick

Um die Implementation für eine nächste Version zu verbessern, könnte man die Genauigkeit des Zeitstempels erhöhen.



### 3.2.4 Log Analyzer

Mit dem WCF Dienst Log Analyzer können die Log Einträge nach allen Log Informationen ausgewertet werden.

In der Message GetLog werden die Auswahlkriterien des Benutzers via Weboberfläche entgegen genommen. Ausser dem LogLevel dürfen die Werte leer sein. Gefiltert wird anschliessend nur nach Kriterien, welche nicht leer sind.

#### 3.2.4.1 Service Contract

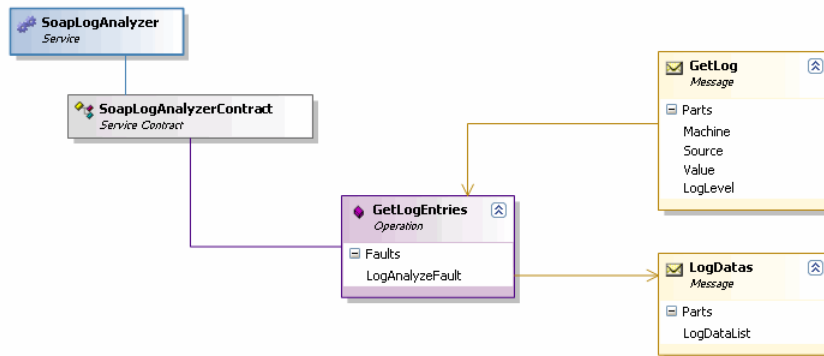


Abbildung 56: SoapLogAnalyzer Service Contract

Die einzige Methode GetLogEntries liest die Kriterien ein und filtert damit die Einträge von der DB. Zurückgegeben wird eine Liste mit Log Einträgen welche diese Kriterien erfüllen.

#### 3.2.4.2 Data Contact

Der Log Analyzer verwendet denselben Data Contract wie der Log Server. Die Beschreibung ist in Kapitel 3.2.3.2 zu finden.

#### 3.2.4.3 Datenbanktabelle

Der Log Analyzer verwendet dieselbe Tabelle wie der Log Server. Die Beschreibung ist in Kapitel 3.2.3.3 zu finden.

#### 3.2.4.4 Ausblick

Als mögliche Erweiterung könnte man in einer folgenden Version die Meldungen zwischen zwei Diensten so darstellen, dass man Dialoge einfach sichtbar machen kann.

### 3.2.5 Weboberfläche

Dem Benutzer steht eine Weboberfläche zur Verfügung um das System zu nutzen. Erstellt wurde die Weboberfläche mit der WCSF. Die Trennung von Anzeige, Anzeigelogik und Daten wird in der WCSF mit dem MVP-Pattern umgesetzt. Dazu gibt es mehrere Gründe:

- Bessere Wartbarkeit
- Automatische Tests mit Test-Framework möglich
- Mehrere Webseiten im selben Modul können unterschiedliche Sichtweisen auf dieselben Daten ermöglichen. Im Controller wird aber derselbe Code verwendet (Code sharing).
- Es wäre möglich anstelle der Webseiten eine andere Ansicht (View) auf die Daten zu implementieren, sprich eine Windows Forms Applikation. Presenter und Controller bleiben aber gleich.
- Modularität

#### Model

Das Model repräsentiert die Daten und Businessfunktionalität (Controller). Der Controller stellt jeweils über die Proxyklassen den Zugriff auf die WCF Dienste sicher.

#### View

Die View ist die visuelle Repräsentation des Models und besteht aus Forms und Controls. Diese Schicht kann durch einen Modultest ersetzt werden um die Funktionalität aus Perspektive der Oberfläche zu prüfen

#### Presenter

Der Presenter stellt die Verbindung zwischen Model und View her und enthält die Präsentationslogik sowie die Prüfung der Input-Daten.

Damit MVP seine eigentlichen Vorteile gegenüber MVC entfalten kann, werden für Modell und Ansicht jeweils Interfaces verwendet. Die Schnittstellen definieren den genauen Aufbau beider Schichten und der Präsentator verknüpft lediglich die Schnittstellen miteinander. Dies gewährleistet die vollständige Austausch- und Wiederverwertbarkeit des Modells und der Ansicht.

In der Abbildung auf der rechten Seite sind neben dem MVP Pattern auch die Arbeitsweise am Beispiel der Verarbeitung von Daten mit Hilfe einer ObjectContainerDataSource ersichtlich. Bei einer Änderung der Daten wird mit einem Event in der View der Event Handler ausgeführt, welcher den Presenter aufruft. Die Daten werden anschliessend durch die Schichten weitergegeben, aktualisiert (CRUD steht für "Create, Read, Update and Delete"), und wieder zurück durch die Schichten bis zur View geleitet, wo die aktualisierten Daten dargestellt werden.

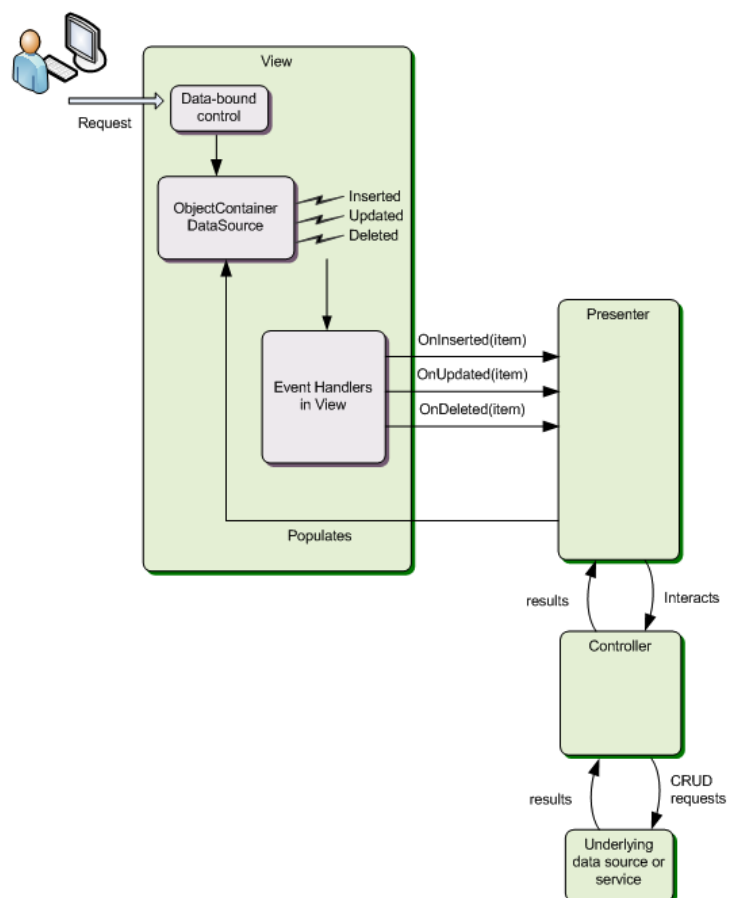


Abbildung 57: MVP Pattern

Die Zusammenhänge zwischen den Klassen im MVP Pattern sind im folgenden Klassendiagramm am Beispiel der Auftragsübersicht ersichtlich:

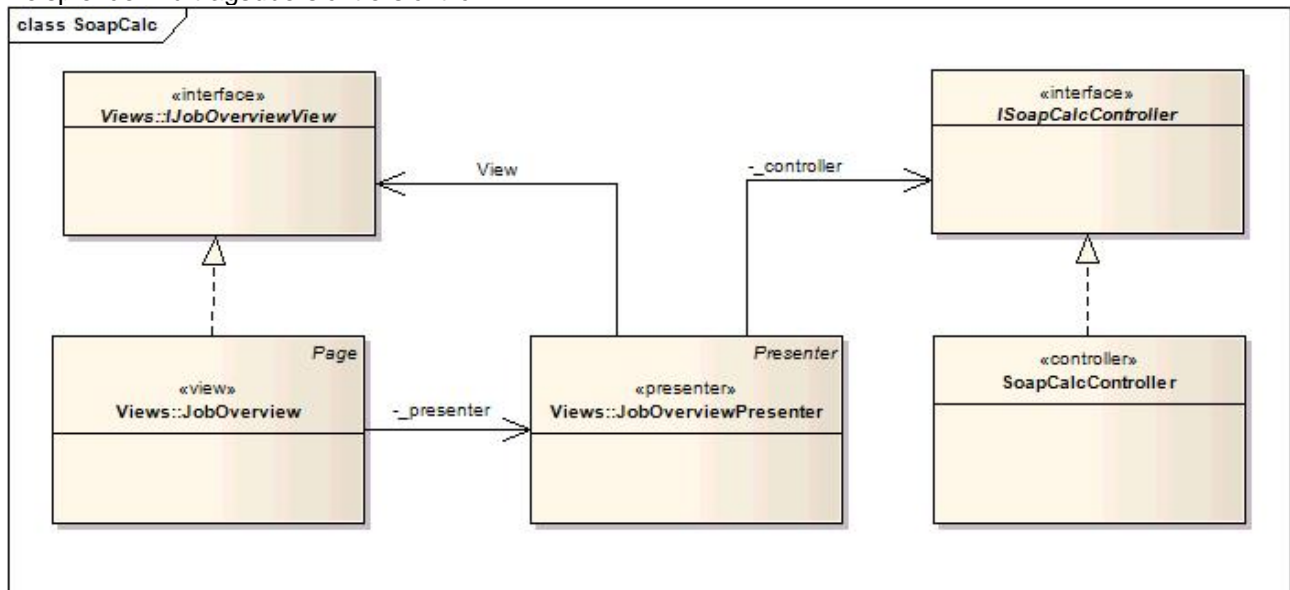


Abbildung 58: MVP Klassendiagramm

### 3.2.5.1 Projektstruktur

Aus dem MVP-Pattern ergibt sich die Struktur der Solution, welche weiter in Module aufgeteilt ist. Auch werden noch zusätzliche Web Controls eingesetzt, um den Bedienkomfort und die Benutzerführung zu verbessern.

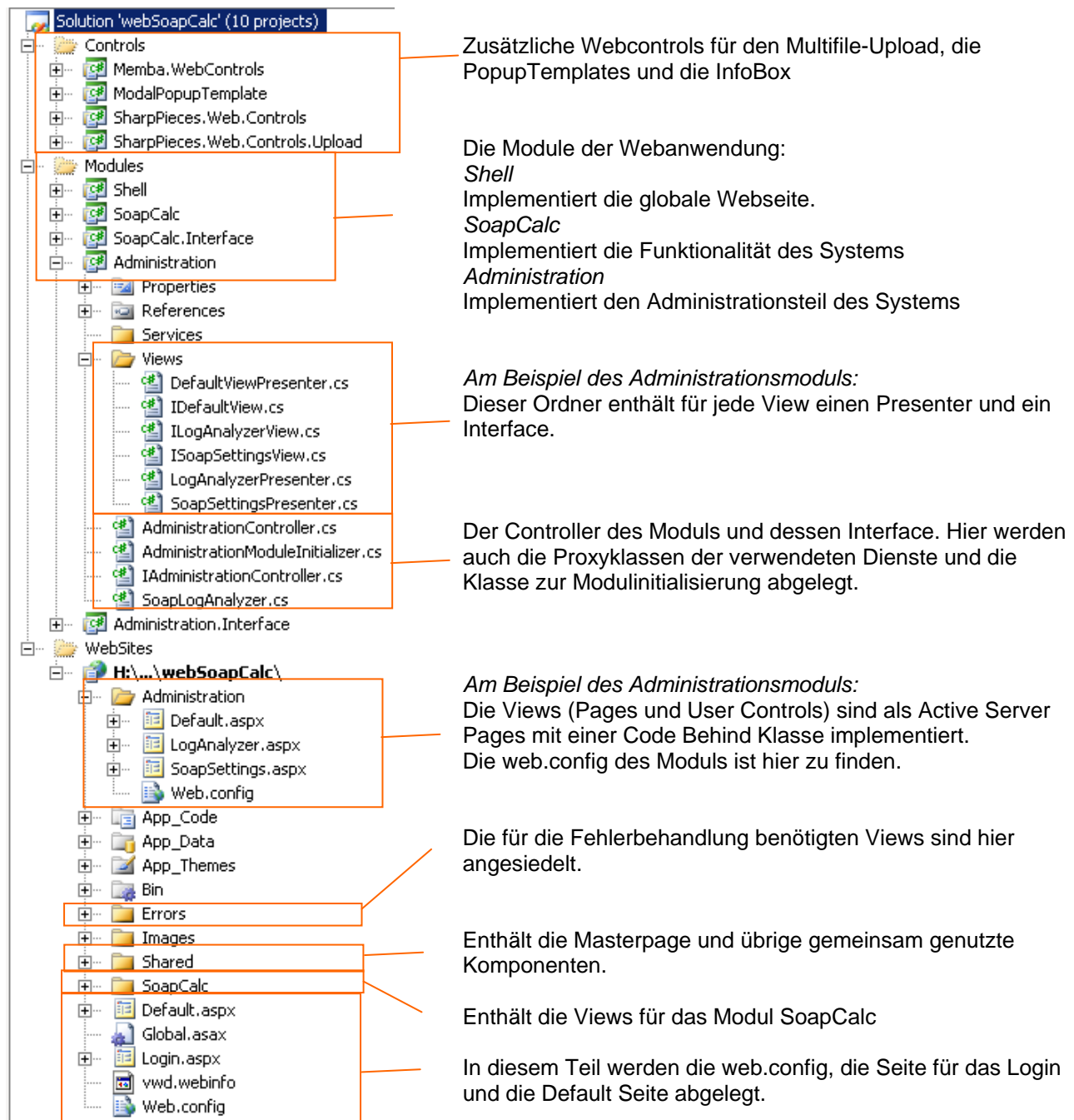


Abbildung 59: Solution der Weboberfläche

Die übrigen Daten wie die css Dateien, die Bilder oder die kompilierten dlls werden in den üblichen Verzeichnissen einer ASP.NET Anwendung abgelegt

### 3.2.5.2 Aufbau der Views

Die Views können in zwei Kategorien eingeteilt werden

- Übersichten
- Detailansichten

#### Übersichten

Übersichten stellen eine Liste von Business Objects tabellarisch dar. Der Aufbau dieser Views ist immer identisch. Verwendet werden dazu die Komponenten GridView sowie eine ObjectContainerDataSource als Datenquelle.

Beim Aufruf der Seite wird auf dem üblichen Weg eine Collection des Business Objects angefordert. Diese wird der ObjectContainerDataSource als Datenquelle zugeordnet.

Als Datenquelle der GridView dient nun die ObjectContainerDataSource. Zudem werden in der GridView alle möglichen Aktionen für jede Zeile angezeigt. Um die Auswahl des Benutzers zu verarbeiten, wird mit Hilfe des DataKeyName der GridView das betroffene Business Object ausgewählt und die passende Methode dazu aufgerufen.

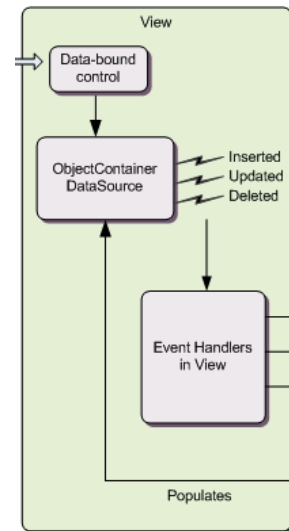


Abbildung 60: View mit ObjectContainerDataSource

#### Detailsansicht

Zur Erfassung und Bearbeitung der Business Objects wird jeweils mit User Controls gearbeitet. Diese stellen ein Formular mit allen Angaben zur Verfügung, welches beim Laden mit einem neuen oder bestehenden Objekt initialisiert wird. Die Angaben können verändert und anschliessend gespeichert werden. Für tabellarische Darstellungen innerhalb des User Controls, wie z.B. die verwendeten Parameter, kam jeweils ein ListView Control zum Einsatz.

User Controls können in eine Page oder in ein Popup–Template integriert werden und sind so wieder verwendbar.

### 3.2.5.3 Zusätzlich verwendete Web Controls

Um den Bedienkomfort und die Benutzerführung zu verbessern werden zusätzliche Web Controls verwendet.

#### Folder Browser

In einem User Control wurden, mit einem Tab Control getrennt, ein Multi File Upload sowie ein File Browser zusammengefasst. Der Multi File Upload setzt JavaScript und Flash ein, um so z.B. den Speicherplatz vor dem Upload zu prüfen und den Verlauf grafisch anzuzeigen. Mit dem File Browser hat man die Möglichkeit, die Dateien des Verzeichnisses herunterzuladen, auszuwählen oder zu löschen.

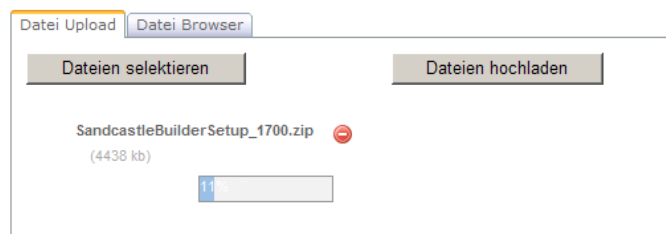


Abbildung 61: File Browser Control

#### InfoBox

Informationen können dem Benutzer mit der InfoBox bequem und verständlich angezeigt werden. Zur Verfügungen stehen Symbole für OK, Information, Warnung und Fehler.



Abbildung 62: Beispiel einer InfoBox

#### Popup Templates

Popup Templates sind AJAX Panels, welche entweder wie eine Message Box oder als Modaler Dialog für User Controls verwendet werden können.

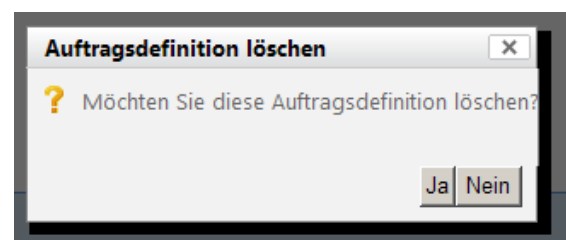


Abbildung 63: Popup Template als MessageBox

### 3.2.5.4 Modul SoapCalc

Das Modul SoapCalc enthält die Funktionalität zur Verwaltung der Aufträge und Auftragsvorlagen, der Auftragsdefinitionen und der Sammelaufträge. Auch bietet das Modul eine Übersicht über die Berechnungsserver.

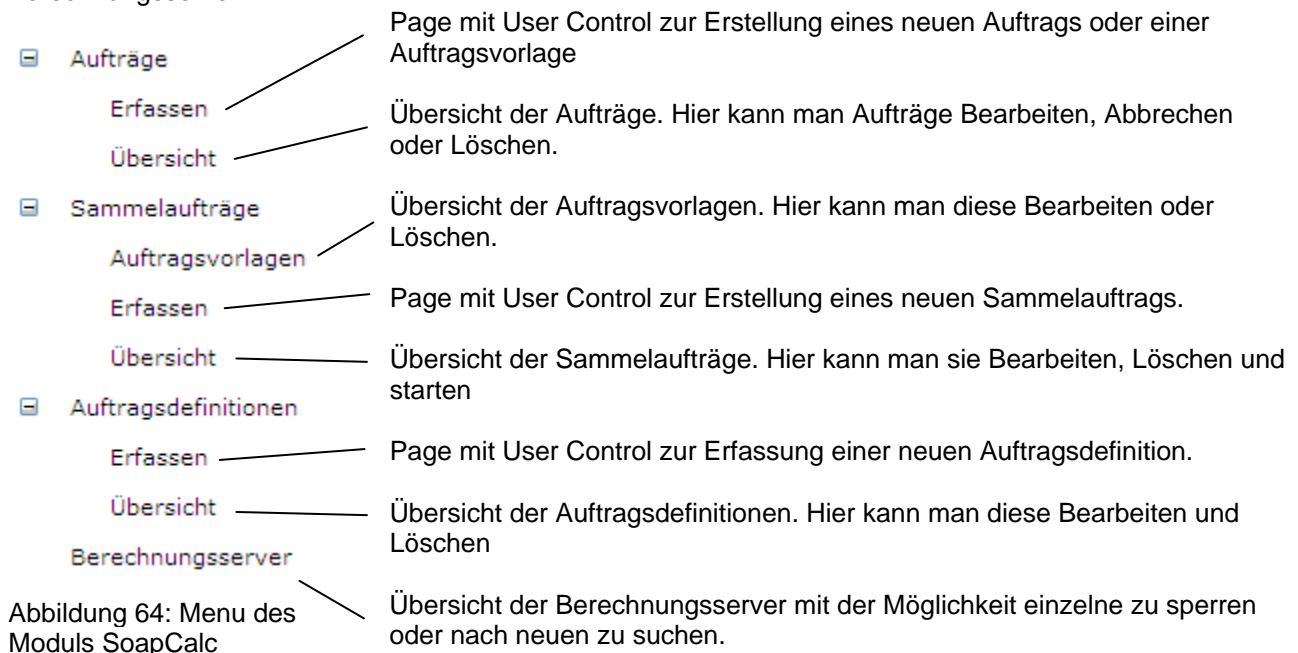


Abbildung 64: Menu des Moduls SoapCalc

Das Screenshot zeigt das User Control 'Sammelauftrag bearbeiten'. Es enthält folgende Elemente:

- Name des Sammelauftrags**: Ein Textfeld mit dem Inhalt 'Abhängigkeiten'.
- Beschreibung**: Ein Textfeld mit dem Inhalt 'Demo'.
- Aufträge**: Eine Tabelle, die die Abhängigkeiten zwischen Vorlagen zeigt.
 

Name	Abhängigkeit 1	Abhängigkeit 2
Vorlage 2	-	-
Vorlage 3	Vorlage 2	-
Vorlage 4	Vorlage 3	-
Vorlage 5	Vorlage 4	-

 Darunter befindet sich ein Dropdown-Menü mit der Auswahl 'Vorlage 1' und ein 'Speichern' Button.
- Statusbar**: Ein grüner Checkmark-Symbol und der Text 'Der Sammelauftrag wurde gespeichert'.

Abbildung 65: User Control des Sammelauftrags

Ein Highlight dieses Moduls ist die Erstellung eines Sammelauftrags. In einem ListView werden die Auftragsvorlagen zusammengestellt und die Abhängigkeiten zueinander definiert. Vor dem Speichern wird im Presenter der View mit einer Graphenbibliothek geprüft, ob keine zirkulären Abhängigkeiten existieren. Erst dann wird das Objekt an den Controller weitergeleitet.

### 3.2.5.5 Modul Administration

Auf diesen Bereich haben nur Benutzer der Gruppe der Administratoren Zugriff. Für die Benutzerverwaltung wird das Active Directory genutzt.

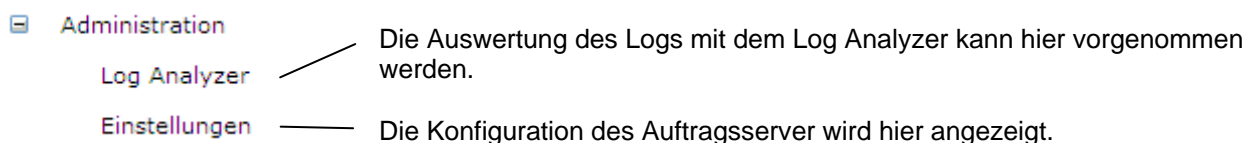


Abbildung 66: Menu des Moduls Administration

Die Ansicht des Log Analyzers:

### Log Analyzer

LogLevel:

Maschine:

Quelle:

Wert:

[Am suchen...](#)

Es werden 1235 Datensätze angezeigt

Maschine	LogLevel	Zeit	Quelle	Wert
SOAPCALC05	Information	03.09.2008 12:46:35	SoapCalc on SOAPCALC05	Verbindung mit Auftragsserver getrennt
SOAPCALC04	Information	03.09.2008 12:46:30	SoapCalc on SOAPCALC04	Verbindung mit Auftragsserver getrennt
SOAPCALC03	Information	03.09.2008 12:46:23	SoapCalc on SOAPCALC03	Verbindung mit Auftragsserver getrennt
SOAPCALC02	Information	03.09.2008 12:46:16	SoapCalc on SOAPCALC02	Verbindung mit Auftragsserver getrennt
SOAPCALC01	Information	03.09.2008 12:27:17	SoapMain CalcServer	CalcServer erstellt
SOAPCALC05	Information	03.09.2008 12:25:38	SoapCalc on SOAPCALC05	Auslastungsreport erstellt
SOAPCALC05	Information	03.09.2008 12:25:33	SoapCalc on SOAPCALC05	Verbunden mit Auftragsserver
SOAPCALC05	Information	03.09.2008 12:25:33		Gestartet...
SOAPCALC05	Information	03.09.2008 12:25:33	SoapCalc LogServer	LogServer ready

Abbildung 67: Log Analyzer

### 3.2.5.6 Errorhandling Webpage

Für die Fehlerbehandlung in der Weboberfläche wird eine eigene Error Page verwendet. In einem Fehlerfall wird der Fehler in der globalen Anwendungsklasse (Global.asax) ausgewertet und auf die Error Page umgeleitet. Fehlermeldungen von den Diensten kommen als Soap Exceptions bei der Weboberfläche an. Die Nachricht der Fehlermeldung wird dann an die globale Anwendungsklasse weitergeleitet.

### 3.2.5.7 Konfiguration

Alle Einstellungen für die Weboberfläche sind in der web.config abgelegt. Zusätzlich konfiguriert wurde hier der Zugriff auf das Active Directory sowie die Zugriffsberechtigungen und das Exception Handling. In den Unterordnern der Module wurden die Einstellungen für die Zugriffsberechtigungen der Module sowie die Angaben für die Nutzung der WCF Dienste eingetragen.

### 3.2.5.8 Ausblick

Als dringende Erweiterung steht bei der Weboberfläche die Überarbeitung des File Handlings an. Das eingesetzte Control erfüllt die Funktionalität, allerdings nur mit einem Workaround. Nach jedem Upload ist der Zustandsverwaltung zurückgesetzt. Hier wurde deshalb eine eigene Zustandsverwaltung mit einem XML File eingesetzt. Als Alternative könnte man den Datei Upload über einen zusätzlichen WCF Dienst vornehmen.

### **3.3 Systemklassen-Diagramm**

Aufgrund der Vielzahl der Klassen wird hier auf eine Darstellung aller Klassen verzichtet. Einzelne interessante Klassen sind in der Systemklassenbeschreibung (Kap. 3.2) erläutert.

### **3.4 Nachrichtenfolge**

Die Nachrichtenfolge innerhalb und zwischen den Systemteilen wird exemplarisch anhand eines Beispiels dargestellt. Diese ist adaptierbar für alle Operationen. Gezeigt wird die Nachrichtenfolge für den Fall, dass der Benutzer einen neuen Auftrag erfasst.



### 3.4.1 Weboberfläche – Auftragsserver

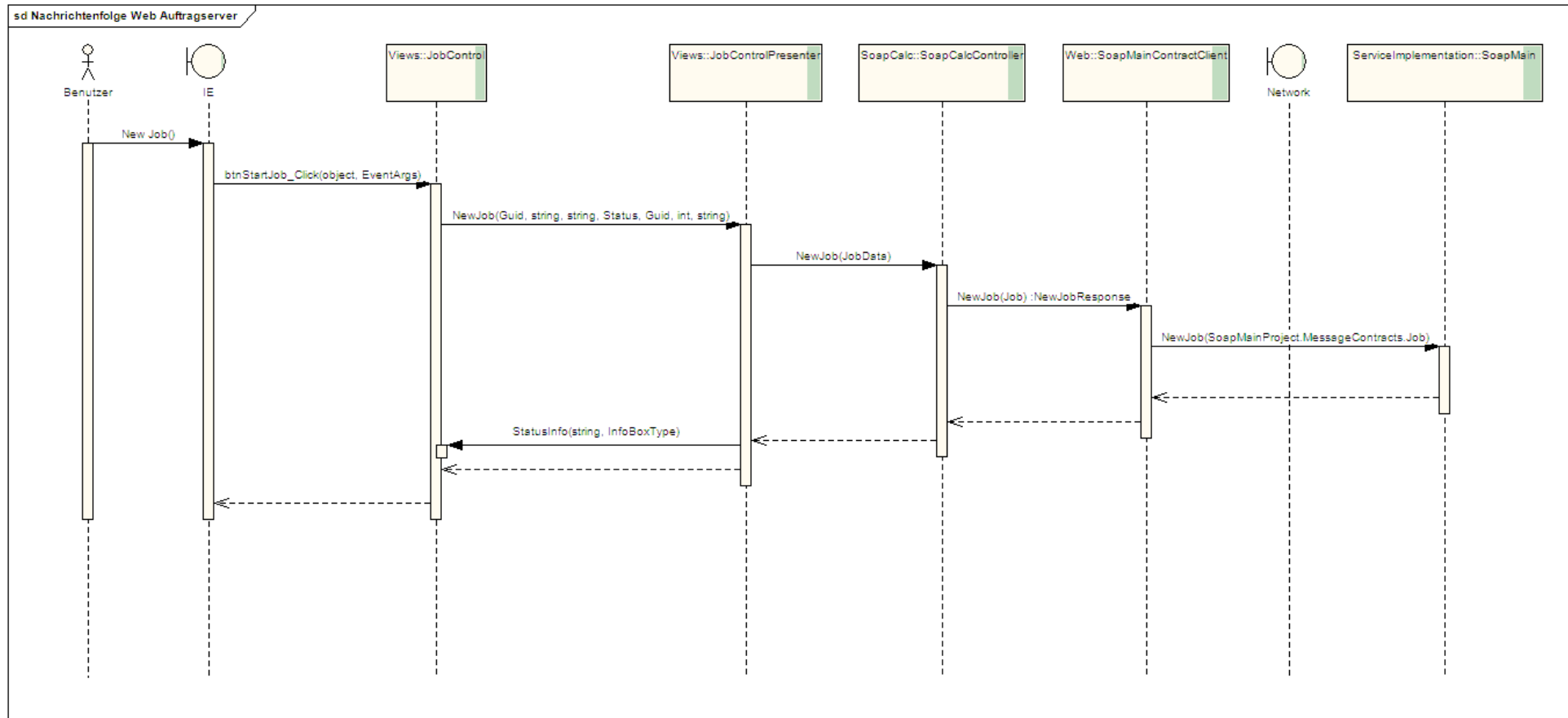


Abbildung 68: Nachrichtenfolge Weboberfläche - Auftragsserver

### 3.4.2 Auftragsserver – Logger

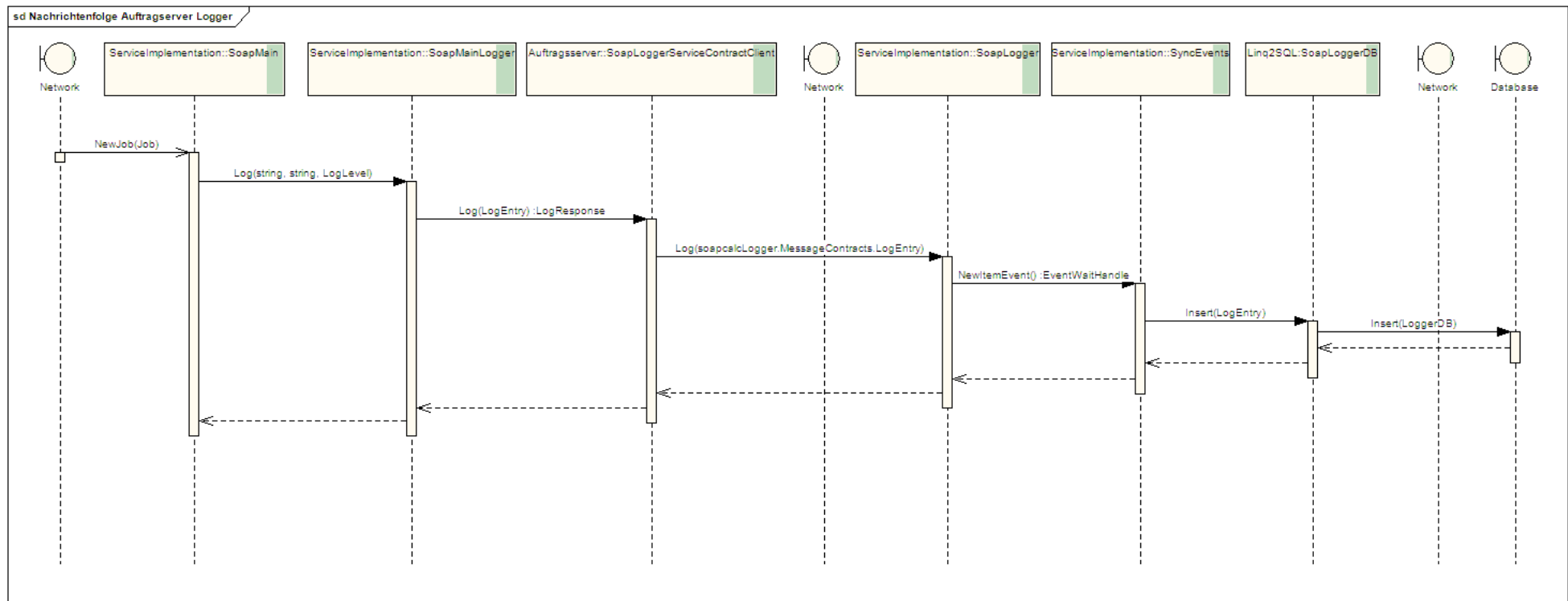


Abbildung 69: Nachrichtenfolge Auftragsserver - Logger

### 3.4.3 Auftragserver - Berechnungsserver

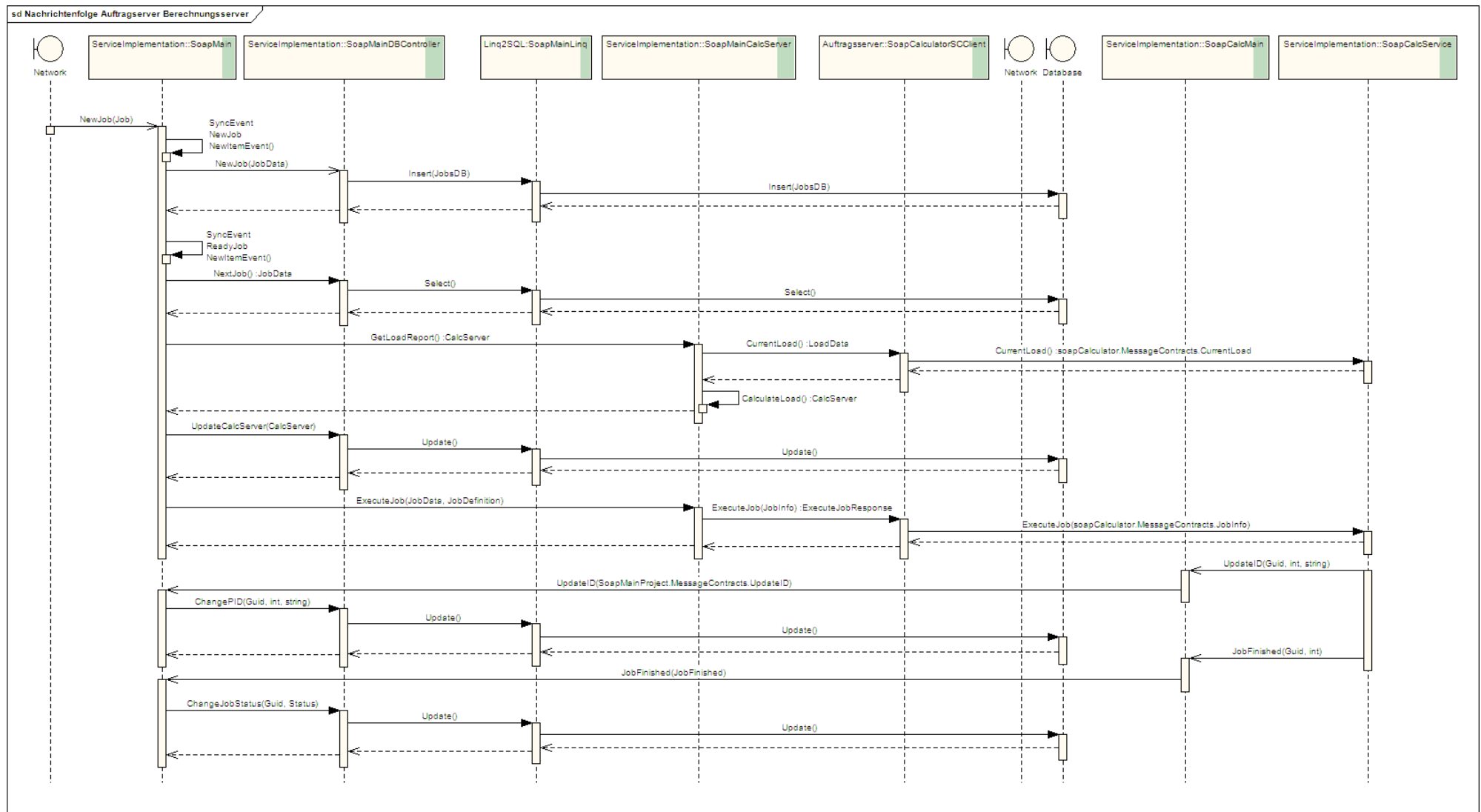


Abbildung 70: Nachrichtenfolge Auftragserver – Berechnungsserver für die Ausführung eines Auftrags

### 3.5 Initialisierung der verschiedenen Objekte

Da die WCF Dienste als Singleton implementiert sind, wird die Initialisierung mit dem ersten Aufruf gestartet. Dies bedeutet, dass beim ersten Start der Weboberfläche die Dienste gestartet werden. Im folgenden Diagramm ist auch die Multiplizität der einzelnen Objekte sichtbar.

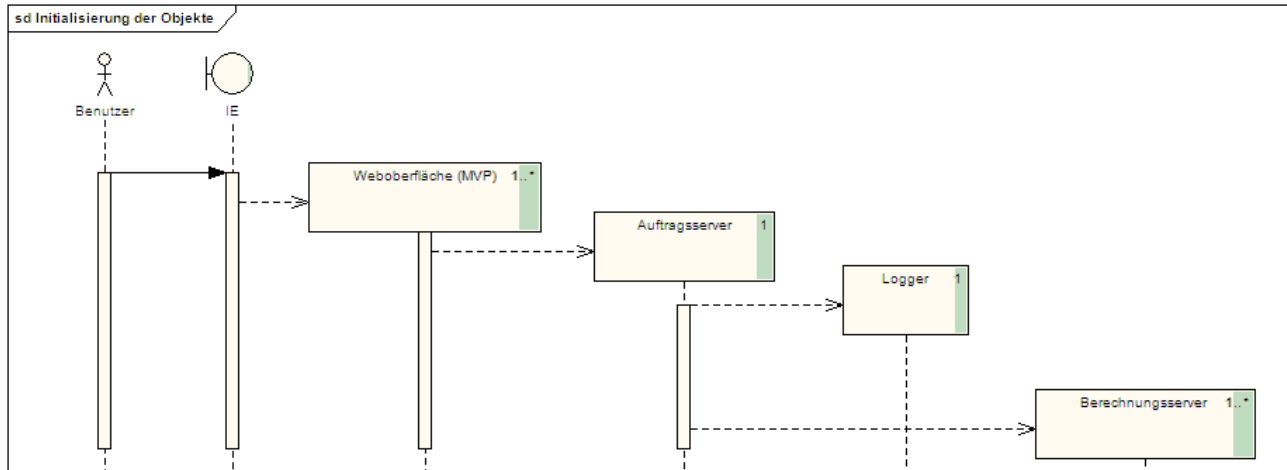


Abbildung 71: Initialisierung der Objekte

## 3.6 Algorithmen

### 3.6.1 Lastindikator

Zur Leistungsbewertung verschiedener Informatiksysteme stellt das Transaction Processing Performance Council standardisierte Benchmarks zur Verfügung. Diese gehen aber für dieses Projekt zu weit, deshalb wurden nur die wichtigsten Performance Indizes in einer eigenen Formel zusammengestellt. Weitere Infos findet man unter <http://www.tpc.org/>

#### 3.6.1.1 Formel zur Berechnung des Lastindikators

$$L = 100 \cdot \left( \frac{\frac{CpuLoad}{100}}{CpuCore} \right) + \frac{CpuFreq}{200} + 2 \cdot \left( \frac{100'000}{Mem} \right) + \frac{1}{User} + (if(FreeDisk < 1GB) then + 100)$$

CpuLoad = Aktuelle CPU Last [%]

CpuCore = Anzahl Cores [1]

CpuFreq = CPU Taktfrequenz [MHz]

Mem = Verfügbarer Arbeitsspeicher [kB]

User = Anzahl User auf dem Server [1]

FreeDisk = Freier Disk Speicher [GB]

#### 3.6.1.2 Betrachtung verschiedener Zustände

Aktuelle Werte						Faktoren inkl. Gewichtung						Lastindikator	Kommentar
Load	Cores	Freq	Mem	User	Disk	Load/CPU	Freq	Mem	User	Disk		Summe	
100	4	2200	100000	2	1	100.00	11.00	2.00	0.50	0.00		113.50	Vergleich des Verhaltens mit untersch. Anzahl Cores
100	2	2200	100000	2	1	100.00	11.00	2.00	0.50	0.00		113.50	
100	1	2200	100000	2	1	100.00	11.00	2.00	0.50	0.00		113.50	
90	4	2200	100000	2	1	90.00	11.00	2.00	0.50	0.00		103.50	Vergleich des Verhaltens mit untersch. Last. Ausloten der Grenze
90	2	2200	100000	2	1	90.00	11.00	2.00	0.50	0.00		103.50	
90	1	2200	100000	2	1	90.00	11.00	2.00	0.50	0.00		103.50	
75	4	2200	100000	2	1	75.00	11.00	2.00	0.50	0.00		88.50	
75	2	2200	100000	2	1	75.00	11.00	2.00	0.50	0.00		88.50	
75	1	2200	100000	2	1	75.00	11.00	2.00	0.50	0.00		88.50	
50	4	2200	100000	2	1	50.00	11.00	2.00	0.50	0.00		63.50	
50	2	2200	100000	2	1	50.00	11.00	2.00	0.50	0.00		63.50	
50	1	2200	100000	2	1	50.00	11.00	2.00	0.50	0.00		63.50	
50	4	1800	100000	2	1	50.00	9.00	2.00	0.50	0.00		61.50	Vergleich des Verhaltens mit untersch. Taktfrequenz
50	2	1800	100000	2	1	50.00	9.00	2.00	0.50	0.00		61.50	
50	1	1800	100000	2	1	50.00	9.00	2.00	0.50	0.00		61.50	
50	4	1800	100000	3	1	50.00	9.00	2.00	0.33	0.00		61.33	Vergleich des Verhaltens mit untersch. Anzahl Usern
50	2	1800	100000	3	1	50.00	9.00	2.00	0.33	0.00		61.33	
50	1	1800	100000	3	1	50.00	9.00	2.00	0.33	0.00		61.33	
50	4	1800	150000	3	1	50.00	9.00	1.33	0.33	0.00		60.67	Vergleich des Verhaltens mit untersch. freiem Arbeitsspeicher
50	2	1800	150000	3	1	50.00	9.00	1.33	0.33	0.00		60.67	
50	1	1800	150000	3	1	50.00	9.00	1.33	0.33	0.00		60.67	
50	4	1800	150000	3	0.5	50.00	9.00	1.33	0.33	100.00		160.67	Vergleich des Verhaltens ohne freien Disk Space
50	2	1800	150000	3	0.5	50.00	9.00	1.33	0.33	100.00		160.67	
50	1	1800	150000	3	0.5	50.00	9.00	1.33	0.33	100.00		160.67	

Ein Berechnungsserver kann neue Aufträge annehmen solange der Lastindikator kleiner als 90 ist. Gewählt wird immer der Berechnungsserver mit dem kleinsten Lastindikator.

## 4 Prototyp

### 4.1 WCF mit Web Service Software Factory

#### 4.1.1 Absicht

Ziel dieses Prototyps ist es, einen WCF Dienst mit Hilfe der Web Service Software Factory (WSSF) zu erstellen. Als Host wird der IIS verwendet. Zusätzlich zum Visual Studio 2008 wird noch die Web Service Software Factory benötigt. Diese muss installiert werden.

Einerseits wird die Funktion der zusätzlich installierten Komponenten geprüft. Auch gilt es, erste Erfahrungen im Umgang mit der WSSF zu machen und abzuklären, ob diese für den Einsatz in diesem Projekt geeignet ist.

Als Vorlage dient ein Logger welcher bereits als Abschlussübung als Web Service erstellt wurde.

Dabei ruft ein Client die Methode eines Services für einen Log Eintrag auf. Der Service nimmt diesen entgegen und speichert ihn in eine Datenbank.

#### 4.1.2 Realisation

Die WSSF integriert sich direkt im Visual Studio. Nach Anlegen einer neuen Solution „Model Project“ vom Typ „Service Factory: Modeling Edition“ kann man beginnen, den Dienst zu modellieren.

##### 4.1.2.1 Service Contract

Zu Beginn wird ein Service Contract erstellt. Die einzelnen Teile (Service, Service Contract, Operation und Messages) können per drag and drop erstellt werden.

Anschließend werden diese konfiguriert und miteinander verbunden. Dies sieht dann so aus:

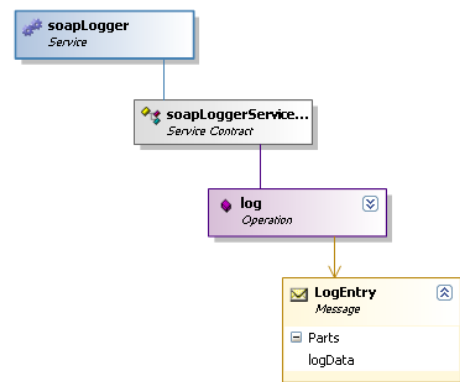


Abbildung 72: Service Contract vom Prototyp Logger

##### 4.1.2.2 Data Contract

Ähnlich wie beim Service Contract werden auch die Teile des Data Contracts per drag and drop erstellt. Verwendet wird in diesem Beispiel zusätzlich zum Data Contract eine Enumeration für den LogLevel.

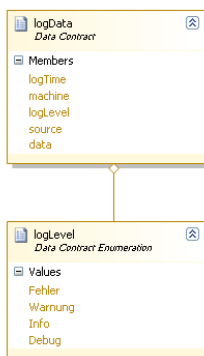


Abbildung 73: Data Contract vom Prototyp Logger

#### 4.1.2.3 Host

Zum Schluss wird der Host konfiguriert. Dazu werden die Service Beschreibungen und die Endpunkte definiert.

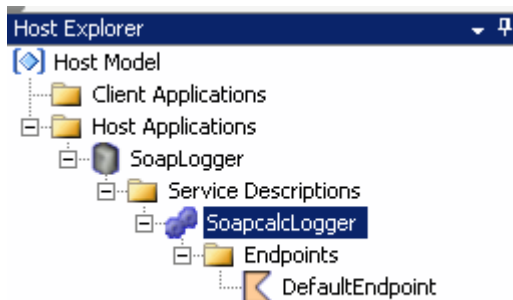


Abbildung 74: Host Explorer

Anschliessend an das Modellieren werden alle Komponenten validiert. Sämtliche benötigten Klassen inklusive entsprechender Konfiguration können automatisch generiert werden. Dabei wird auch die benötigte Struktur angelegt.

Die eigentliche Implementation erfolgt nun in partiellen Klassen welche an die generierten partiellen Klassen anschliessen.

Zur Erstellung eines Proxy Objekts mit passender Konfigurationsdatei für den WCF Client wird das Tool svcutil.exe verwendet. Dieses sammelt seine Angaben aus dem wsdl des Dienstes.

Beispiel eines Aufrufs:

```
svcutil.exe http://localhost:2556/SoapcalcLogger.Host/soapcalcLogger.svc?wsdl
```

Im Client wird alles gegen diese Proxy Klasse programmiert. Bei Änderungen am Dienst muss diese wieder aktualisiert werden.

### 4.1.3 Testergebnisse und Erkenntnisse

Der Prototyp funktioniert gut. Die Fleissarbeit modellieren und konfigurieren gestaltet sich sehr einfach. Die Erwartungen an die Software Factory wurden erfüllt.

Die WSSF funktioniert wie erwartet sehr gut und ist einfach zu bedienen. Die Konfiguration ist einfach und übersichtlich, was bei den vielen Möglichkeiten und damit Parametern der WCF von Vorteil ist. Die Einarbeitung durch Erstellung eines Dienstes gelingt in einer kurzen Zeit. Die Möglichkeit den Dienst zu modellieren und anschliessend den benötigten Code zu generieren ist sehr komfortabel.

Ein Nachteil der WSSF ist, dass diese von sich aus keine Callbacks unterstützt.

Durch die Tatsache, dass mit der WCF die Unterstützung einer grossen Auswahl an Übertragungswegen und -protokollen, einschliesslich SOAP auf dem konfigurativen Weg gewährleistet ist und keinen Hostcode geschrieben oder für IIS bereitstellen werden muss, steht einem Einsatz in diesem Projekt nichts mehr im Wege.

Die Performance wurde hier nicht bewertet, da diese für das Projekt nicht ausschlaggebend ist. Die Hauptzeit wird beim Berechnen der einzelnen Aufträge benötigt.

## 4.2 LINQ und die WCSF

### 4.2.1 Absicht

Die Log Einträge aus dem ersten Prototyp sollen mit LINQ aus der Datenbank ausgelesen und als Liste in einer Webseite präsentiert werden. Zum Einsatz kommen soll neben LINQ die WCSF zur Erstellung einer Weboberfläche für den Benutzer.

### 4.2.2 Realisation

Die WCSF integriert sich direkt im Visual Studio. Nach Anlegen einer neuen Solution vom Typ Web Client Development wird zuerst die benötigte Struktur (Model, View, Presenter) automatisch erstellt. Anschliessend stehen Assistenten zur Verfügung, um diese einzelnen Views hinzuzufügen.

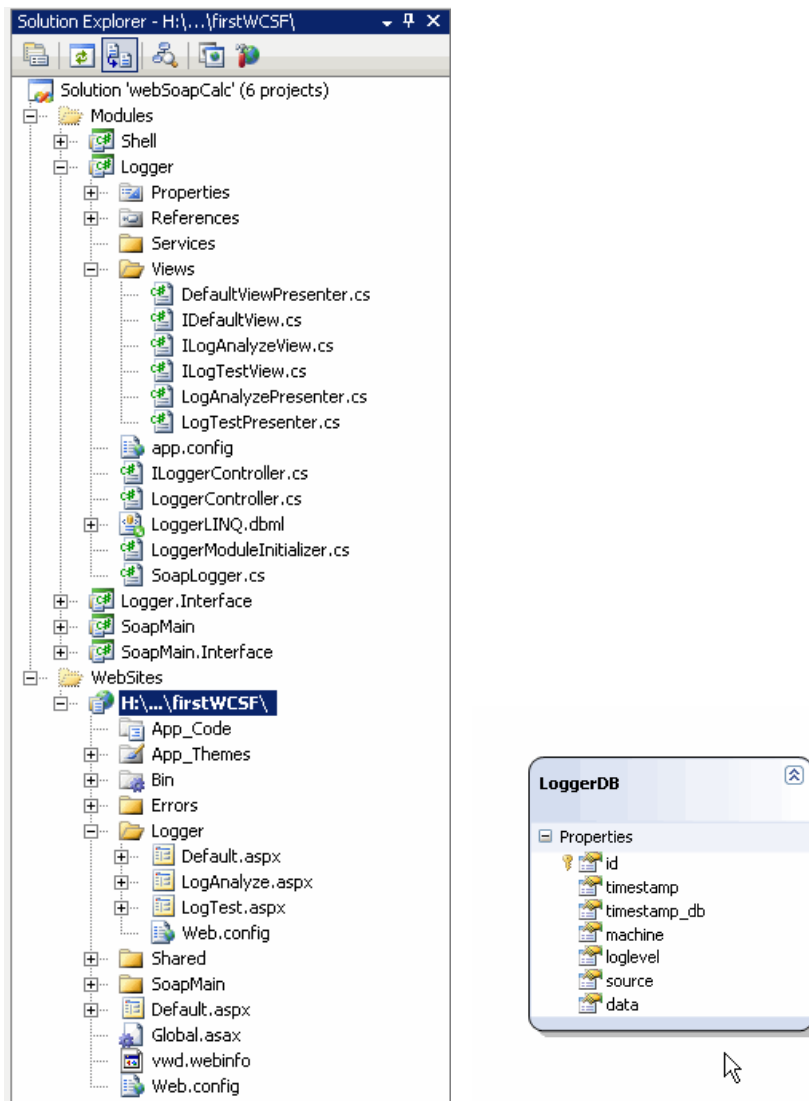


Abbildung 75: Struktur des WCSF Prototyps

Abbildung 76: Tabelle im LINQ to SQL DataContext

Um LINQ zu verwenden fügt man ein neues Item vom Typ „LINQ to SQL Classes“ hinzu. Die benötigten Tabellen können nun direkt per Drag and Drop aus dem Server Explorer in die Ansicht gezogen werden. Die für den Zugriff benötigten DataContext Klassen, auf welche mit LINQ zugegriffen werden kann, werden automatisch erstellt.

Die Log Einträge wurden über den Controller mit LINQ aus der Datenbank gelesen und in eine ObjectContainerDataSource geladen. Diese wurde als Tabelle in der View ausgegeben.



Eine weitere View stellt zwei Textboxen zum Eintragen eines neuen Log Eintrags zur Verfügung. Der Inhalt dieser wurde durch die Schichten an den Controller durchgegeben, welcher über den WCF Prototypen einen Eintrag in die DB vornahm.

### 4.2.3 Testergebnisse und Erkenntnisse

Durch die saubere Trennung der WCSF in drei Schichten ist der Web Client übersichtlich strukturiert. Die Möglichkeit den View durch Unit Tests zu ersetzen bietet sich für grössere Projekte an. Neue Views oder Module können sehr einfach hinzugefügt werden.

Das Darstellen von Daten aus einer Datenbank gelingt mit dem eingesetzten Control ObjectContainerDataSource in sehr kurzer Zeit.

Die Daten wurden mit LINQ aus der Datenbank abgefragt. Der Vorteil von LINQ ist, dass es in die Sprache integriert ist und auch mit Objekten arbeitet. Diese können weiterverwendet werden. Die Syntax ist sehr ähnlich wie der von SQL. Auch kann bei Bedarf ein SQL Statement direkt abgesetzt werden.

Als Nachteil sehe ich, dass die Objekte aus der WSSF und aus LINQ to SQL nicht miteinander kompatibel sind, sprich bei Einsatz beider Hilfsmittel müssen die Objekte umgepackt werden.

Ein weiterer Nachteil von LINQ ist zudem, dass Objekte für Änderungen zuerst ausgelesen, dann geändert und zum Schluss wieder zurückgeschrieben werden müssen. Als Vergleich zu einem Update oder einem Delete Statement in SQL ist dies eher umständlich.

## 5 Testfälle

Dieses Kapitel zeigt die definierten Testfälle sowie deren Erfolg. Die einzigen nicht erfolgreichen Testfälle werden gemäss den Änderungen des Pflichtenhefts nicht mehr benötigt (Nummer 23, 24 und 31), bzw. die Anforderungen wurden nicht implementiert. Dies betrifft zwei Anforderungen mit Priorität 3, die Nummer 33 und 35. Angewendet wurden die Testfälle während der Entwicklung einzelner Komponenten, als Abschlusstest auf dem Entwicklungssystem sowie nach der Installation auf dem produktiven System.

### 5.1 Auftragsserver

Testfall	Szenario	Beschreibung	Anforderung	Ausgangslage	Erwartetes Resultat	Erfolg	FB
101	Logger initialisieren	Proxy-Objekt für Logger erstellen und Starteintrag loggen	40	Laden des Konstruktors vom Auftragsserver	Proxy-Objekt erstellt, Logeintrag vorgenommen	ok	ok
102	DB Controller initialisieren	Objekt für DB Controller erstellen	10.3	Laden des Konstruktors vom Auftragsserver	Objekt für DB Controller erstellt	ok	ok
103	Erster Berechnungsserver initialisieren	Proxy-Objekt für Berechnungsserver erstellen, DB Eintrag updaten	13	Laden des Konstruktors vom Auftragsserver	Proxy-Objekt erstellt, Tabelle CalcServerDB geleert, Berechnungsserver mit aktueller Auslastung in DB eingetragen	ok	ok
104	Timer initialisieren	Timer anlegen mit Angaben aus dem Config-File, Event registrieren	13.2	Laden des Konstruktors vom Auftragsserver	Timer erstellt und Event registriert	ok	ok
105	Thread "Auftrag vorbereiten" starten	Neuen Thread "Auftrag vorbereiten" erstellen und starten	10, 10.1	Laden des Konstruktors vom Auftragsserver	Neuen Thread "Auftrag vorbereiten" gestartet	ok	ok
106	Thread "Auftrag verteilen" starten	Neuen Thread "Auftrag verteilen" erstellen und starten	10, 13	Laden des Konstruktors vom Auftragsserver	Neuen Thread "Auftrag verteilen" gestartet	ok	ok
107	Neuer Auftrag in Queue schreiben	Ein Auftrag mit Status Neu in die Queue schreiben. Anschliessend wird ein Event "Neuer Job" gefeuert	10, 10.1	Weboberfläche löst neuen Auftrag aus	Auftrag in Queue, SyncEvent für Thread "Auftrag vorbereiten" gefeuert.	ok	ok
108	Auftrag, welcher neu keine Abhängigkeiten mehr hat, in Queue schreiben	Ein Auftrag mit Status Waiting kann in die Queue geschrieben werden. Anschliessend wird ein Event "Neuer Job" gefeuert.	12, 12.1	Die Abhängigkeiten eines Auftrags wurden abgeschlossen. Der Auftrag kann jetzt verteilt werden.	Auftrag in Queue, SyncEvent für Thread "Auftrag vorbereiten" gefeuert	ok	ok
109	Daten für Auftrag vorbereiten	Kopieren der benötigten Daten (Definition und Input)	10.3, 10.4	Ein Auftrag ohne Abhängigkeiten wird vorbereitet zur Verteilung	Ein Ordner mit den benötigten Daten ist angelegt	ok	ok

110	Status eines Auftrags ändern	Neuen Status eines Auftrags in DB eintragen	10	Der Status eines Auftrags muss vom Ablauf her geändert werden	Neuer Status in DB eingetragen	ok	ok
111	Event "Auftrag verteilen" feuern aus Thread "Auftrag vorbereiten"	Der Event signalisiert dem Thread "Auftrag verteilen", das ein neuer Job zur Verteilung bereitsteht	10	Ein Auftrag ist bereit zur Verteilung	Event gefeuert	ok	ok
112	Anzahl Aufträge mit Status Bereit abfragen	Im Thread "Auftrag verteilen" wird für den Ablauf die Anzahl bereiter Aufträge benötigt	13	Der Thread "Auftrag verteilen" wurde durch einen Event gestartet	Anzahl Aufträge mit Status Bereit bekannt	ok	ok
113	Nächsten Auftrag auswählen	Der nächste bereite Auftrag wird ausgewählt unter Berücksichtigung der Startzeit und der Priorität	13.3	Anzahl Aufträge mit Status Bereit ist grösser als 0	Auftrag zur Verteilung ausgewählt	ok	ok
114	Auslastungsreport für alle angemeldeten Berechnungsserver anfordern	Für jeden angemeldeten Berechnungsserver den Auslastungsreport abfragen, die Auslastung berechnen und die DB aktualisieren	15.2	Ein Auftrag ist bereit zur Verteilung und der Thread "Auftrag verteilen" wurde durch einen Event gestartet	Die Tabelle für die Berechnungsserver wurde aktualisiert.	ok	ok
115	Berechnungsserver mit der tiefsten Last auswählen	Der aktuell am wenigsten ausgelastete Berechnungsserver, der nicht gelockt ist, wird ausgewählt	13.1	Die Tabelle für die Berechnungsserver wurde aktualisiert.	Der Berechnungsserver mit der tiefsten Last ist bestimmt	ok	ok
116	Auftrag an Berechnungsserver verteilen	Ein Auftrag wird auf dem ausgewählten Berechnungsserver gestartet	13.1	Der Berechnungsserver mit der tiefsten Last ist bestimmt	Auftrag an Berechnungsserver übergeben	ok	ok
117	Timer starten	Der Timer wird gestartet und feuert nach abgelaufener Zeit einen "Auftrag verteilen" Event	13.2	Mind. 1 Auftrag ist bereit zur Verteilung und alle Berechnungsserver sind ausgelastet oder gesperrt	Timer gestartet, Event gefeuert	ok	ok
118	Timer stoppen	Der Timer wird gestoppt	13.2	Keine Aufträge sind bereit zur Verteilung	Timer gestoppt	ok	ok
119	ID und Server eintragen	Entgegennahme der Prozess ID und des Berechnungsservernamens, Eintrag der Daten in die DB	10	Auftrag in Bearbeitung	Daten in der DB eingetragen	ok	ok

120	Auftragsabschluss	Die Abschlussmeldung vom Berechnungsserver wird entgegengenommen. Anhand des Exit-Codes wird der Erfolg ermittelt. Abhängigkeiten in wartenden Aufträgen werden gesucht und gelöscht. Aufträge, welche neu keine Abhängigkeit mehr haben werden in die Queue eingetragen und ein Event "Neuer Job" gefeuert.	10	Auftrag beendet	Auftragsstatus in DB aktualisiert, Aufträge ohne weitere Abhängigkeit in Queue geschrieben.	ok	ok
121	Auftrag abbrechen	Ein Auftrag wird abgebrochen. Status in DB ändern und allenfalls Auftrag auf Berechnungsserver terminieren. Folgeaufträge (Sammelauftrag auch abbrechen)	14, 14.1, 14.2, 14.3	Weboberfläche sendet Abbruchbefehl eines bestehenden Auftrag	Auftrag abgebrochen, Status in der DB aktualisiert.	ok	ok
122	Auftrag verändern	Ein Auftrag, welcher noch nicht verteilt wurde, kann verändert werden.	10.2	Weboberfläche sendet Änderung eines bestehenden Auftrag	Auftrag wurde geändert, die Daten im Ordner aktualisiert	ok	ok
123	Auftrag löschen	Ein Auftrag wird gelöscht. Falls dieser in Bearbeitung ist, wird er zuerst abgebrochen	10.6	Weboberfläche sendet Löschauftrag für bestehenden Auftrag	Auftrag wurde in der DB gelöscht, der Datenordner gelöscht	ok	ok
124	Liste mit Aufträgen erstellen	Eine Liste aller Aufträge erstellen.	10.5	Weboberfläche fordert Liste der Aufträge an.	Liste mit Aufträgen erstellt	ok	ok
125	Auftragsdefinition erstellen	Eine neue Auftragsdefinition kann erstellt werden. Die Daten werden in die DB eingetragen	11, 11.1, 11.3, 11.4	Weboberfläche sendet neue Auftragsdefinition	Auftragsdefinition wurde in die DB eingetragen, ein Ordner mit den benötigten Daten angelegt	ok	ok
126	Auftragsdefinition ändern	Eine Auftragsdefinition kann verändert werden	11.2, 11.3, 11.4	Weboberfläche sendet Änderung einer bestehenden Auftragsdefinition	Auftragsdefinition wurde in der DB geändert, die Daten im Ordner aktualisiert	ok	ok
127	Auftragsdefinition löschen	Eine Auftragsdefinition kann gelöscht werden	11.6	Weboberfläche sendet Löschauftrag für bestehende Auftragsdefinition	Auftragsdefinition wurde in der DB gelöscht, der Ordner wurde gelöscht	ok	ok

128	Liste mit Auftragsdefinitionen erstellen	Eine Liste aller Auftragsdefinitionen erstellen.	11.5	Weboberfläche fordert Liste der Auftragsdefinitionen an.	Liste mit Auftragsdefinitionen erstellt	ok	ok
129	Sammelauftrag erstellen	Ein neuer Sammelaufrag kann erstellt werden. Die Daten werden in die DB eingetragen	12, 12.1	Weboberfläche sendet neuen Sammelaufrag	Sammelauftrag wurde in die DB eingetragen	ok	ok
130	Sammelauftrag verändern	Ein Sammelaufrag kann verändert werden.	12.2	Weboberfläche sendet Änderung eines bestehenden Sammelaufrag	Sammelauftrag wurde in der DB geändert	ok	ok
131	Sammelauftrag löschen	Ein Sammelaufrag wird gelöscht.	12.3	Weboberfläche sendet Löschaufrag für bestehenden Sammelaufrag	Sammelauftrag wurde aus der DB gelöscht	ok	ok
132	Berechnungsserver anmelden	Dieser Testfall ist durch eine Änderung des Pflichtenhefts nicht mehr erforderlich.	15	Neuer Berechnungsserver verfügbar	Berechnungsserver eingetragen		
133	Berechnungsserver abmelden	Ein Berechnungsserver steht dem Auftragsserver nicht mehr zur Verfügung. Dieser wird aus der Liste gelöscht. Der Eintrag in der DB wird gelöscht.	15	Berechnungsserver nicht mehr verfügbar	Berechnungsserver gelöscht	ok	ok
134	Berechnungsserver testen	Ein Dummy-Auftrag durchläuft das System und wird auf einem ausgewählten Berechnungsserver berechnet.	15.3	Weboberfläche löst Berechnungsserver Test aus	Erfolgsstatus wird der Weboberfläche übergeben	ok	ok
135	Berechnungsserver sperren	Ein Berechnungsserver kann gesperrt werden, Eintrag in DB	15	Ein Berechnungsserver wird vom Weboberfläche gesperrt.	Eintrag in DB vorgenommen	ok	ok
136	Berechnungsserver entsperren	Ein Berechnungsserver kann wieder entsperrt werden, Eintrag in DB	15	Ein Berechnungsserver wird vom Weboberfläche entsperrt.	Eintrag in DB vorgenommen	ok	ok

137	Liste mit Berechnungsserver erstellen	Für jeden angemeldeten Berechnungsserver den Auslastungsreport abfragen, die Auslastung berechnen und die DB aktualisieren. Anschliessend wird eine Liste aller Berechnungsserver erstellt.	15.1, 15.2	Weboberfläche fordert Liste der Berechnungsserver an.	Liste mit Berechnungsserver erstellt	ok	ok
138	Berechnungsserver suchen	Zur Verfügung stehende, nicht angemeldete Berechnungsserver werden gesucht	17	Weboberfläche löst Suche aus	Gefundene Berechnungsserver angemeldet	ok	ok
139	Liste mit Einstellungen erstellen	Eine Liste aller Parameter der Application Properties wird erstellt	16	Weboberfläche fordert Liste der Einstellungen an.	Liste mit Einstellungen erstellt	ok	ok
140	Einstellung ändern	Dieser Testfall ist durch eine Änderung des Pflichtenhefts nicht mehr erforderlich.	16.1	Weboberfläche ändert den Wert für ein Property	Property geändert		

## 5.2 Berechnungsserver

Testfall	Szenario	Beschreibung	Anforderung	Ausgangslage	Erwartetes Resultat	Erfolg	FB
201	Logger initialisieren	Proxy-Objekt für Logger erstellen und Starteintrag loggen	40	Laden des Konstruktors vom Berechnungsserver	Proxy-Objekt erstellt, Logeintrag vorgenommen	ok	ok
202	Auftragsserver initialisieren	Proxy-Objekt für Auftragsserver erstellen	31	Laden des Konstruktors vom Berechnungsserver	Proxy-Objekt erstellt	ok	ok
203	Anmeldung am Auftragsserver	Dieser Testfall ist durch eine Änderung des Pflichtenhefts nicht mehr erforderlich.	31	Laden des Konstruktors vom Berechnungsserver	Berechnungsserver angemeldet		
204	Abmeldung vom Auftragsserver	Der Berechnungsserver meldet sich am Auftragsserver ab	32	Herunterfahren des Dienstes	Berechnungsserver abgemeldet	ok	ok
205	Entgegennehmen eines Auftrags	Der Berechnungsserver nimmt einen Auftrag entgegen	30, 30.1	Auftragsserver übergibt Auftrag an Berechnungsserver	Auftrag entgegengenommen	ok	ok
206	Verarbeiten des Auftrags	Ein neuer Thread wird erstellt. Mit den Angaben des Auftrags wird ein neuer Prozess gestartet	30.2	Auftrag entgegengenommen	Verarbeitung gestartet	ok	ok

207	Melden der PID und des Servers	Die Process ID des gestarteten Prozesses und der Name des Berechnungsservers werden an den Auftragsserver gemeldet	30.3	Verarbeitung gestartet	PID und Servername gemeldet	ok	ok
208	Abschluss des Auftrags melden	Im Thread wird auf das Ende der Verarbeitung gewartet. Anschliessend wird der Exit-Code an den Auftragsserver gemeldet und der Thread beendet	30.4	Verarbeitung abgeschlossen	Exit-Code gemeldet	ok	ok
209	Ablehnung eines Auftrags	Dieser Testfall ist durch eine Änderung des Pflichtenhefts nicht mehr erforderlich.	33	Benötigte Anwendung nicht vorhanden	Auftrag abgelehnt		
210	Auslastungsreport erstellen	Abfragen der aktuellen Systemwerte, Erstellung eines Reports	34	Auftragsserver fordert Auslastungsreport an	Auslastungsreport erstellt	ok	ok
211	Abbrechen eines Auftrags	Abbrechen eines Auftrags anhand der PID	14	Auftragsserver fordert Abbruch eines Prozesses	Prozess abgebrochen	ok	ok
212	Performancefaktor berechnen	Dieser Testfall ist durch eine Änderung des Pflichtenhefts nicht mehr erforderlich.	35	Berechnung wird angefordert	Performancefaktor berechnet		

### 5.3 Weboberfläche

Testfall	Szenario	Beschreibung	Anforderung	Ausgangslage	Erwartetes Resultat	Erfolg	FB
301	Auftragsserver initialisieren	Proxy-Objekt für Auftragsserver erstellen	20	Initialisieren des Controllers	Proxy-Objekt erstellt	ok	ok
302	Erfassen eines neuen Auftrags	Erfassen eines neuen Auftrags	21	Benutzer erfasst neuen Auftrag	Auftrag an Auftragsserver übergeben	ok	ok
303	Ändern eines bestehenden Auftrags	Ändern eines bestehenden Auftrags	21	Benutzer ändert bestehenden Auftrag	Auftragsänderung an Auftragsserver übergeben	ok	ok
304	Löschen eines bestehenden Auftrags	Löschen eines bestehenden Auftrags	21	Benutzer löscht bestehenden Auftrag	Auftragslöschung an Auftragsserver übergeben	ok	ok
305	Auftrag abbrechen	Abbruch eines bestehenden Auftrags	14	Benutzer bricht Auftrag ab	Abbruch an Auftragsserver übergeben	ok	ok
306	Anzeigen der Auftragsübersicht	Anzeigen der Auftragsübersicht, wird beim Auftragsserver angefordert	20	Benutzer fordert Auftragsübersicht an	Übersicht wird angezeigt	ok	ok

307	Erfassen einer neuen Auftragsdefinition	Erfassen einer neuen Auftragsdefinition	20	Benutzer erfasst neue Auftragsdefinition	Neuer Auftrag an Auftragsserver übergeben	ok	ok
308	Ändern einer bestehenden Auftragsdefinition	Ändern einer bestehenden Auftragsdefinition	20	Benutzer ändert bestehende Auftragsdefinition	Auftragsdefinitionsänderung an Auftragsserver übergeben	ok	ok
309	Löschen einer bestehenden Auftragsdefinition	Löschen einer bestehenden Auftragsdefinition	20	Benutzer löscht bestehende Auftragsdefinition	Auftragsdefinitionslöschung an Auftragsserver übergeben	ok	ok
310	Anzeigen der Auftragsdefinitionsübersicht	Anzeigen der Auftragsdefinitionsübersicht, wird beim Auftragsserver angefordert	20	Benutzer fordert Auftragsdefinitionsübersicht an	Übersicht wird angezeigt	ok	ok
311	Erfassen eines neuen Sammelauftrags	Erfassen eines neuen Sammelauftrags	20	Benutzer erfasst neuen Sammelauftrag	Sammelauftrag an Auftragsserver übergeben	ok	ok
312	Ändern eines bestehenden Sammelauftrags	Ändern eines bestehenden Sammelauftrags	20	Benutzer ändert bestehenden Sammelauftrag	Sammelauftragsänderung an Auftragsserver übergeben	ok	ok
313	Löschen eines bestehenden Sammelauftrags	Löschen eines bestehenden Sammelauftrags	20	Benutzer löscht bestehenden Sammelauftrag	Sammelauftragslöschung an Auftragsserver übergeben	ok	ok
314	Anzeigen der Sammel-auftragssübersicht	Anzeigen der Sammelauftragssübersicht, wird beim Auftragsserver angefordert	20	Benutzer fordert Sammelauftragssübersicht an	Übersicht wird angezeigt	ok	ok
315	Anzeigen der Berechnungsserverübersicht	Anzeigen der Berechnungsserverübersicht, wird beim Auftragsserver angefordert	22	Benutzer fordert Berechnungsserverübersicht an	Übersicht wird angezeigt	ok	ok
316	Sperren eines Berechnungsservers	Ein Berechnungsserver soll für neue Aufträge gesperrt werden	15	Benutzer sperrt Berechnungsserver	Sperrung an Auftragsserver übergeben	ok	ok
317	Entsperren eines Berechnungsservers	Ein gesperrter Berechnungsserver soll für neue Aufträge entsperrt werden	15	Benutzer entsperrt Berechnungsserver	Entsperrung an Auftragsserver übergeben	ok	ok



318	Testen eines Berechnungsservers	Ein Dummy-Auftrag durchläuft das System und wird auf einem ausgewählten Berechnungsserver berechnet.	15.3	Benutzer löst Berechnungsserver Test aus	Erfolgsstatus wird dem Benutzer angezeigt	ok	ok
319	Proxy-Objekt für Log Analyzer erstellen	Proxy-Objekt für Log Analyzer erstellen	16.2	Laden des Konstruktors vom Berechnungsserver	Proxy-Objekt erstellt	ok	ok
320	Auswertung des Logs	Auswertung des Logs mit dem Log Analyzer	16.2, 50	Benutzer fordert LogDaten mit Kriterien an	Übersicht wird angezeigt	ok	ok
321	Anzeigen der Einstellungen	Die Properties des Auftragsservers werden angezeigt	23	Benutzer fordert Properties an	Übersicht wird angezeigt	ok	ok
322	Ändern einer Einstellung	Dieser Testfall ist durch eine Änderung des Pflichtenhefts nicht mehr erforderlich.	23	Benutzer ändert Property	Neuer Wert an an Auftragsserver übergeben		
323	Erfassen eines neuen Benutzers	Dieser Testfall ist durch eine Änderung des Pflichtenhefts nicht mehr erforderlich.	24	Administrator erfasst neuen Benutzer	Benutzer angelegt		
324	Ändern eines bestehenden Benutzers	Dieser Testfall ist durch eine Änderung des Pflichtenhefts nicht mehr erforderlich.	24	Administrator ändert bestehenden Benutzer	Benutzer mutiert		
325	Löschen eines bestehenden Benutzers	Dieser Testfall ist durch eine Änderung des Pflichtenhefts nicht mehr erforderlich.	24	Administrator löscht bestehenden Benutzer	Benutzer gelöscht		
326	Anzeigen der Benutzersübersicht	Dieser Testfall ist durch eine Änderung des Pflichtenhefts nicht mehr erforderlich.	24	Administrator fordert Benutzersübersicht an	Übersicht wird angezeigt		

## 5.4 Logger

Testfall	Szenario	Beschreibung	Anforderung	Ausgangslage	Erwartetes Resultat	Erfolg	FB
401	DB Thread starten	Neuen DB Thread erstellen und starten	42	Laden des Konstruktors vom Logger	DB Thread gestartet	ok	ok
402	Event "Neuer Logeintrag" registrieren	SyncEvent für neuen Log Eintrag registrieren	41	Laden des Konstruktors vom Logger	Event registriert	ok	ok
403	Fehler in Textfile eintragen	Fehler in Textfile eintragen	40	Fehler in Verarbeitung	Fehler in Textfile geloggt	ok	ok
404	Log Eintrag entgegennehmen und neuen Event feuern	Log Eintrag in Queue eintragen und SyncEvent feuern	41	Neuer Log Eintrag übergeben	Log Eintrag in Queue eingetragen und SyncEvent gefeuert	ok	ok
405	Log Eintrag in DB schreiben	Log Eintrag in DB schreiben	42, 60	SyncEvent gefeuert	Log Eintrag in DB gespeichert	ok	ok

## 5.5 Log Analyzer

Testfall	Szenario	Beschreibung	Anforderung	Ausgangslage	Erwartetes Resultat	Erfolg	FB
501	Liste mir Log Einträgen erstellen	Liste mir Log Einträgen erstellen, gefiltert nach den Kriterien in der Anfrage	50	Weboberfläche fordert Log Entries an	Liste wurde erstellt	ok	ok

## 6 Schlussbemerkungen

Mit dieser Diplomarbeit konnte ich mich noch einmal eingehend mit dem Ablauf eines Softwareprojekts auseinandersetzen, den gelernten Stoff der letzten Semester einsetzen und das theoretisch erlernte Fachwissen prüfen. Das Resultat ist aus meiner Sicht gelungen und ich freue mich, der Enkom Inventis AG ein gut laufendes System zu übergeben.

Die Motivation den Herausforderungen zu begegnen hat während der gesamten Arbeit nie nachgelassen. Die aufgetauchten Probleme haben mir immer wieder von neuem den Ansporn gegeben um nach Lösungen zu suchen. Entsprechend gross war auch der Lerneffekt während der Arbeit, nicht zuletzt auch durch den Einsatz von mir vorgängig nicht bekannten Technologien wie der WCF oder LINQ.

Die Highlights der Arbeit waren für mich neben den neuen Technologien der WCF und LINQ auch der Einsatz der Software Factories, welche das modellieren der gewünschten Anforderungen auf hohem Abstraktionsniveau zulassen. Auch von der Architektur des Systems, vom modularen Aufbau und dem Einsatz verschiedener Patterns (z.B. MVP oder Fassadenpattern), bin ich sehr überzeugt. Durch den Einsatz von Open Source Bibliotheken, beispielsweise der Zip Bibliothek, der Graphen Bibliotheken zur Prüfung von zirkulären Abhängigkeiten oder der zusätzlichen Web Controls konnte die Funktionalität ohne grossen Mehraufwand deutlich verbessert werden.

Der Terminplan wurde bei jedem gesetzten Milestone eingehalten. Unterschätzt wurde jedoch die Zeit um Webseiten mit ASP.NET so zu gestalten dass sie einerseits funktionell und einfach sind, andererseits optisch ansprechend erscheinen. Der Einsatz von AJAX Komponenten steigert den Benutzerkomfort, ist aber mit einem erhöhten Entwicklungsaufwand verbunden. Die Erwartungen an den angekündigten Nachfolger Silverlight sind entsprechend hoch. Durch den Einsatz der Software Factories und von Open Source Bibliotheken jedoch konnte ein grosser Teil dieses Mehraufwands wieder kompensiert werden.

Die Entwicklung von Verteilten Systemen erfordert einen Mehraufwand zur Verwaltung der Konfiguration, für das Deployment und das Testen des Systems. Der Einsatz von Unit Tests würde hier für letzteres sicher Verbesserungen bringen. Allerdings decken diese auch nicht alle Bereiche ab, wie. z.B. der Test von User Interfaces.

Bis zum Schluss haben mich die Unterschiede zwischen dem Entwicklungswebserver des Visual Studios und des IIS im produktiven System beschäftigt, dies obwohl ich das System einen Monat vor der Abgabe zum ersten Mal auf dem produktiven System getestet habe. Nach dem Deployment sind Fehler aufgetreten, welche auf dem Entwicklungswebserver nicht zu beobachten gewesen sind.

Eine Aussage über die Reduktion der Berechnungsdauer kann zurzeit noch nicht gemacht werden, weil die benötigten Skripts und Batches noch nicht zur Verfügung stehen. Der Verwaltungsaufwand zur Verteilung der Testaufträge benötigt jedoch nur wenige Sekunden. Dies lässt erwarten, dass die angestrebte Verbesserung erreicht wurde.

Als sehr konstruktiv habe ich die Zusammenarbeit mit dem Experten und dem Betreuer empfunden. Für die geleistete Unterstützung und die gute Zusammenarbeit möchte ich mir herzlich bei den Herren Wenger und Bigler bedanken!

## 7 Anhang

Anhang A:	Pflichtenheft
Anhang B:	Quellenverzeichnis
Anhang C:	Abbildungsverzeichnis
Anhang D:	Informationen zum Deployment
Anhang E:	Zugriff auf den Auftragsserver

---

## **Anhang A: Pflichtenheft**



**Berner Fachhochschule**  
Technik und Informatik

**Software-Schule Schweiz**  
**Diplomarbeit MAS-06-01.06**

# SOAPCALC

## **Pflichtenheft**

Klasse MAS-IT-06-01, März 2008

### **Abstract:**

Im Rahmen der Mobilfunknetz- Planung und Optimierung fallen häufig rechenintensive Datenverarbeitungen an wie z.B. die Berechnung der Abdeckung der Schweizer Bevölkerung mit dem Mobilfunkdienst HSPA oder die Verarbeitung von georeferenzierten Daten (Messungen, GIS Layer usw.). Die Idee von SOAPCALC ist es, diese Berechnungen auf verschiedene Server zu verteilen um die Verarbeitungszeit zu verkürzen und die Abläufe zu automatisieren. Das System ist aber frei konfigurierbar für beliebige rechenintensive Datenverarbeitungen.

For the radio network planning and optimization, computationally intensive data processings are often required, e.g. the calculation of the coverage of the Swiss population with the service HSPA or the processing of georeferencing data (Measures, GIS layer etc.). The goal of SOAPCALC is to distribute these calculations on different servers in order to reduce the calculation time and to automate the workflow. However, the system is freely configurable for any computationally intensive calculations.

### **Schlüsselwörter:**

Verteiltes Rechnen, SOA

### **Student:**

Michael Berger, Mönchstrasse 33, 3600 Thun  
michael.berger@embe.ch, 079 506 07 69

### **Betreuer:**

Stefan Bigler, INVENTiS AG, Worbstrasse 225, 3073 Gümligen  
stefan.bigler@inventis.ch, 031 950 42 31

### **Experte:**

Rolf Wenger, INFOBRAIN AG, Obere Zollgasse 75, 3072 Ostermundigen  
rolf.wenger@infobrain.com, 031 544 22 22

# Inhaltsverzeichnis

1	Einführung .....	3
1.1	Zweck .....	3
1.2	Produktumfang .....	3
1.3	Definitionen und Abkürzungen .....	3
1.4	Glossar .....	4
1.5	Einsatzgebiet .....	4
1.6	Abgrenzungen .....	4
2	Gesamtübersicht .....	5
2.1	Umfeld .....	5
2.1.1	Benutzer-Schnittstelle .....	5
2.1.2	Kommunikations-Schnittstelle .....	5
2.1.3	Software-Schnittstelle .....	5
2.1.4	Hardware .....	5
2.2	Produkt-Funktionalität .....	6
2.2.1	Schema .....	6
2.2.2	Auftragsserver .....	6
2.2.3	Berechnungsserver .....	6
2.2.4	Log Server .....	6
2.2.5	User Interface .....	6
3	Spezifische Anforderungen .....	7
3.1	Funktionen .....	7
3.1.1	Auftragsserver .....	7
3.1.2	Berechnungsserver .....	7
3.1.3	Log Server .....	7
3.1.4	User Interface .....	8
3.2	Mengengerüst .....	8
3.3	Datenbasis .....	9
3.3.1	Datenbank .....	9
3.3.2	Auftragsdefinitionen .....	9
3.3.3	Aufträge .....	9
3.3.4	Sammelauftrag .....	10
3.3.5	Auslastungsreport .....	10
3.4	Sicherheit .....	11
3.4.1	Komponenten .....	11
3.4.2	Benutzer .....	11
4	Tests .....	11
5	Zeitplan .....	12
5.1	Milestones .....	12
6	Anhang .....	13
6.1	Anforderungen .....	13

# 1 Einführung

## 1.1 Zweck

Berechnungsaufgaben sollen auf verschiedene Server verteilt werden um die Verarbeitungszeit zu verkürzen und die Abläufe zu automatisieren. Die Software verwaltet diese Aufgaben und kontrolliert die Verteilung oder führt die Aufträge aus. Dem Benutzer steht eine Weboberfläche zur Verfügung, um den Service zu konfigurieren und Aufträge zu erfassen.

## 1.2 Produktumfang

SOAPCALC umfasst folgende Komponenten

- Auftragsserver
- Berechnungsserver
- Log Server
- User Interface

## 1.3 Definitionen und Abkürzungen

### Auftragsserver

Der Auftragsserver verwaltet die Aufträge und verteilt diese an die Berechnungsserver.  
Siehe auch Kapitel 3.1.1

### Berechnungsserver

Ein Berechnungsserver nimmt die Aufträge entgegen und verarbeitet diese.  
Siehe auch Kapitel 3.1.2

### Log Server

Zentraler Log Server, der allen Komponenten zur Verfügung steht.  
Siehe auch Kapitel 3.1.3

### Auftragsdefinition

Eine Auftragsdefinition definiert die Berechnung, welche ausgeführt werden soll.  
Siehe auch Kapitel 3.3.2

### Auftrag

Ein Auftrag beinhaltet alle benötigten Angaben und Daten, um von einem Berechnungsserver verarbeitet zu werden. Ein Auftrag ist die kleinste mögliche Einheit.  
Siehe auch Kapitel 3.3.3

### Sammelauftrag

Mehrere Aufträge werden in einem Sammelauftrag zusammengefasst. Die Aufträge können in Abhängigkeit eines anderen Auftrags sein. Der Sammelauftrag wird auf dem Auftragsserver verarbeitet und als einzelne Aufträge an die Berechnungsserver weitergeleitet.  
Siehe auch Kapitel 3.3.4

### Auslastungsreport

Der Auslastungsreport beinhaltet Informationen zur aktuellen Auslastung eines Berechnungsservers.  
Siehe auch Kapitel 3.3.5

### SOAPCALC

Projektname, setzt sich aus dem zu verwendenden Protokoll SOAP und CALC für Berechnen zusammen.



## 1.4 Glossar

GIS	- Geographisches Informationssystem
SOA	- service oriented architecture
GSM	- Global System for Mobile Communications
UMTS	- Universal Mobile Telecommunications System
HSPA	- High Speed Packet Access
EDGE	- Enhanced Data Rates for GSM Evolution
IIS	- Internet Information Server der Firma Microsoft (Web- und FTP- Server)
FTP	- File Transfer Protocol
SOAP	- Simple Object Access Protocol
WCF	- Windows Communication Foundation

## 1.5 Einsatzgebiet

SOAPCALC wird im Umfeld der Mobilfunknetz- Planung und Optimierung eingesetzt. Aufträge sind z.B. die Berechnung der Abdeckung der Schweizer Bevölkerung mit dem Mobilfunkdienst HSPA oder die Verarbeitung von georeferenzierten Daten (Messungen, GIS Layer usw.).

Grundsätzlich kann aber SOAPCALC für beliebige Aufgaben eingesetzt werden, wo Berechnungen über die Kommandozeile oder per Skript gestartet werden können. So z.B. auch für die Umwandlung von WAV Dateien nach mp3.

Bsp. Für den Einsatz in der Mobilfunknetzplanung:

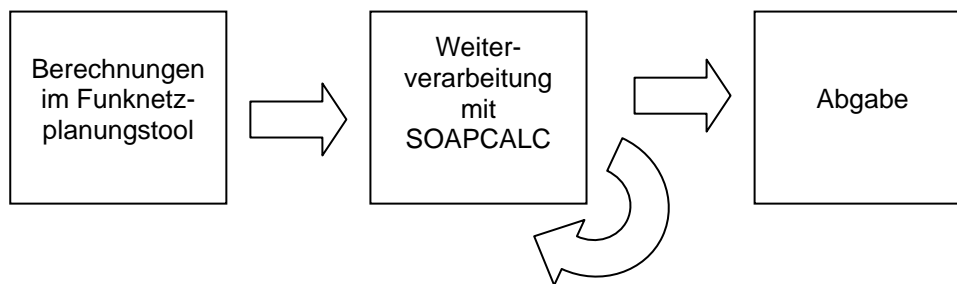


Abbildung 1: Prinzipschema von SOAPCALC

## 1.6 Abgrenzungen

Die Aufgabe umfasst die Entwicklung des Systems SOAPCALC. Dies umfasst einen Auftragsserver, Berechnungsserver, einen Log Server und ein User Interface.

Davon ausgenommen ist die Entwicklung der Skripts und Batches, welche die eigentlichen Aufträge darstellen.

Weiter muss das System nur innerhalb eines privaten Netzwerkes betrieben werden können und nicht zwingend über Netzwerkgrenzen hinweg (Firewallkonfiguration).

Die Installation der Server kann manuell vorgenommen werden, d.h. es ist keine automatische Installation z.B. eines Berechnungsserverdienstes bei der ersten Anmeldung vorgesehen.

Alle Berechnungsserver werden als gleichwertig behandelt. Eine manuelle Auswahl des Berechnungsservers ist nicht möglich.

## 2 Gesamtübersicht

### 2.1 Umfeld

#### 2.1.1 Benutzer-Schnittstelle

Dem Benutzer steht eine Weboberfläche zur Verfügung, um das System zu nutzen. Diese kann mit einem Webbrowser angesprochen werden.

Die Benutzer des Systems sind technisch versiert (Fachspezialisten, Ingenieure)

#### 2.1.2 Kommunikations-Schnittstelle

Als Kommunikationsprotokoll soll ein bestehendes offenes Protokoll verwendet werden (z.B. SOAP).

Der Benutzer greift per http oder allenfalls https auf die Weboberfläche zu.

Für die Übertragung der benötigten Daten ist das Protokoll zu evaluieren.

#### 2.1.3 Software-Schnittstelle

Als Zielplattform stehen Server mit Windows Server 2003 und .NET 3.5 zur Verfügung. Auf den jeweiligen Servern ist IIS installiert.

#### 2.1.4 Hardware

Als Hardware stehen mehrere IBM xSeries 335 Server zur Verfügung. Diese sind mit jeweils 2 Dual Core Prozessoren und 4GB RAM ausgerüstet und sind über ein Fast Ethernet (100MBit/s) - Netzwerk miteinander verbunden.



Abbildung 2: Hardware

Die Performance eines einzelnen vorhandenen physikalischen Servers reicht gut aus um einen Auftrags- und einen Log Server gleichzeitig zu betreiben.

Um eine optimale Performance für die Berechnungsserver zu erreichen, sollte auf einem physikalischen Server nur ein Berechnungsserver betrieben werden. Die vorhandenen Server sind für einen Berechnungsserver als Minimalkonfiguration zu betrachten.

## 2.2 Produkt-Funktionalität

### 2.2.1 Schema

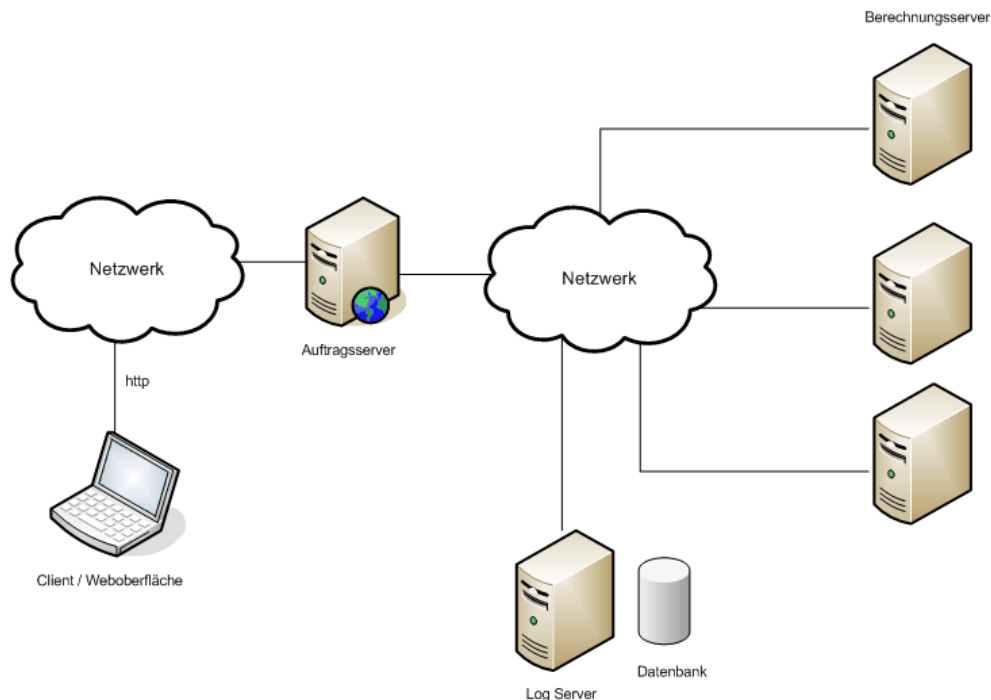


Abbildung 3: Systemschema

### 2.2.2 Auftragsserver

Der Auftragsserver stellt die Hauptkomponente des Systems dar. Hier werden die Aufträge verwaltet und an die Berechnungsserver verteilt. Auch wird die Logik für die Weboberfläche zur Verfügung gestellt.

### 2.2.3 Berechnungsserver

Ein Berechnungsserver nimmt die Aufträge entgegen und verarbeitet diese. Die Anzahl der Berechnungsserver ist variabel (1..n).

### 2.2.4 Log Server

Ein zentraler Log Server, steht allen Komponenten zur Verfügung.

### 2.2.5 User Interface

Dem Benutzer steht eine Weboberfläche zur Verfügung, um das System zu nutzen. Aufgeteilt ist diese Oberfläche in 3 Hauptteile:

- Aufträge verwalten
- Übersicht Berechnungsserver
- Administrationsbereich (Passwortgeschützt)

## 3 Spezifische Anforderungen

### 3.1 Funktionen

#### 3.1.1 Auftragsserver

Die Funktionen des Auftragsservers kann in 3 Bereiche unterteilt werden

##### 3.1.1.1 Auftragsverwaltung

Mit der Auftragsverwaltung können neue Aufträge aus Auftragsdefinitionen erfasst werden.

Es ist zu evaluieren, wie die Aufträge abgespeichert werden. In einem Verzeichnis werden die benötigten Daten abgelegt (siehe auch Kapitel 3.3.3).

Die Aufträge werden, nach Auswertung der Auslastung der Berechnungsserver, an die Berechnungsserver verteilt oder verweilen in den Warteschlangen, bis sie ausgeliefert werden können. Dabei wird die Priorität des Auftrags berücksichtigt.

Aufträge in Bearbeitung können jederzeit abgebrochen werden.

##### 3.1.1.2 Berechnungsserver verwalten

Die Berechnungsserver können sich am Auftragsserver an- oder abmelden. Es kann auch nach aktiven Berechnungsservern gesucht werden. Der aktuelle Auslastungsreport kann angefordert werden.

Optional kann für die Berechnungsserver ein Performancefaktor (siehe Kap. 3.3.5) gerechnet werden.

##### 3.1.1.3 Funktionalität für Weboberfläche

Die Logik für die Weboberfläche wird vom Auftragsserver zur Verfügung gestellt. Dies umfasst eine Auftragsübersicht inkl. sämtlicher Angaben zu einem Auftrag.

Neue Aufträge, Auftragsdefinitionen und Sammelaufträge können erstellt werden.

Die Angaben zu den angemeldeten Berechnungsservern können in einer Übersicht angezeigt werden.

Nicht angemeldete aber aktive Berechnungsserver können gesucht werden.

#### 3.1.2 Berechnungsserver

Beim Start meldet sich der Berechnungsserver am Auftragsserver an, bzw. beim Beenden ab.

Der Berechnungsserver nimmt die Aufträge entgegen und startet diese. Beim start wird die Prozess-ID des Prozesses, der den Auftrag ausführt, dem Auftragsserver zurückgegeben.

Aufträge können nur entgegengenommen werden, wenn der Berechnungsserver nicht schon zu stark ausgelastet ist. Dies ist abhängig von der Anzahl Cores und deren Auslastung (hardwareabhängig) sowie dem verfügbaren Arbeitsspeicher. Ein Auslastungsreport kann erstellt werden, welche diese Information beinhaltet.

Ein Berechnungsserver kann einen Auftrag ablehnen, wenn z.B. ein benötigtes Zusatzprogramm wie MapInfo nicht installiert ist.

Nach Abschluss des Auftrags wird eine Meldung an den Auftragsserver gesendet.

#### 3.1.3 Log Server

Der Log Server besteht aus zwei Teilen:

##### 3.1.3.1 Logger

Ein zentraler Logger steht allen Komponenten von SOAPCALC zur Verfügung. Er nimmt einen Log Eintrag entgegen und schreibt diesen anschliessend in die Datenbank.

Folgende Information wird geloggt:

- Zeitstempel
- Sender
- Log-Level (Fehler, Warnung, Betrieb, Debug)
- Quelle
- Text/Daten

### 3.1.3.2 Log Analyzer

Mit dem Log Analyzer können die Einträge nach allen Informationen ausgewertet werden. Diese Funktionalität steht im Admin Bereich der Weboberfläche zur Verfügung.

## 3.1.4 User Interface

Die Weboberfläche ist in 3 Hauptteile aufgeteilt. Benutzer ohne Administratorenrechte haben Zugriff auf die Funktionalität aus den Kapiteln 3.1.4.1 und 3.1.4.2. Administratoren zusätzlich auch auf die Funktionalität aus Kapitel 3.1.4.3.

### 3.1.4.1 Aufträge verwalten

Auftragsdefinitionen können erstellt oder verändert werden. Aufträge können aus bestehenden Auftragsdefinitionen erfasst werden.

Eine Übersicht stellt die aktuellen Aufträge und deren Status dar. Hier können die Aufträge abgebrochen werden.

Sammelaufträge können erstellt werden.

### 3.1.4.2 Übersicht Berechnungsserver

Eine Übersicht stellt die aktuell angemeldeten Berechnungsserver mit dazugehöriger Information des Auslastungsreports dar.

Ein einzelner Berechnungsserver kann hier getestet werden. Nicht angemeldete Berechnungsserver können gesucht werden.

Optional kann die Berechnung eines Performancefaktors (siehe Kap. 3.3.5) gestartet werden.

### 3.1.4.3 Administrationsbereich (Passwortgeschützt)

Auf diesen Bereich haben nur berechtigte Benutzer Zugriff.

Die Konfiguration des Auftragsserver kann hier vorgenommen werden (Prioritäten, Verzeichnisse usw.). Allenfalls ist dazu ein Neustart des Auftragsserver notwendig.

Eine Benutzerverwaltung erlaubt das Administrieren der Benutzer.

Die Auswertung des Logs kann hier vorgenommen werden.

## 3.2 Mengengerüst

Für den Betrieb von SOAPCALC werden folgende Komponenten benötigt:

- 1 Auftragsserver
- 1 Weboberfläche
- 1 Log Server
- 1..n Berechnungsserver
- 

Zusammenstellung der aktuellen Situation pro Monat für die einzelnen Zyklen

Zyklus	reinen Rechenzeiten	Produzierte GIS Layer	Benötigte Aufträge
UMTS	3-4 Tage	ca. 100	ca. 300
GSM	2-3 Tage	ca. 80	ca. 240
Spezielle Aufträge	2-3 Tage	ca. 20	ca. 100

Da die Rechenzeit von der Anzahl Server abhängig ist, ergibt sich für die verbesserte Rechenzeit folgende Formel

$$Rechenzeit_{Neu} = Rechenzeit_{Alt} \cdot \frac{1}{(Server_{Anzahl} - 1)}$$

Die Anzahl der Aufträge pro Berechnungsserver ist hardwareabhängig.

Die Datenmenge einzelner Aufträge beträgt insgesamt bis ca. 100MB, welche auf mehrere Files verteilt sind.

## 3.3 Datenbasis

### 3.3.1 Datenbank

Die Datenbank beinhaltet Tabellen für den Logger mit den korrespondierenden Spalten. Allenfalls werden für die Auswertung Views auf diese Tabellen benötigt.

Je nach gewähltem Authentifizierungsverfahren werden auch Tabellen für die Benutzer benötigt.

Je nach gewählter Speicherart der Aufträge werden auch Tabellen für den Betrieb des Auftragservers benötigt.

### 3.3.2 Auftragsdefinitionen

Eine Auftragsdefinition definiert die Berechnung, welche ausgeführt werden soll. Dies beinhaltet

- Eindeutige ID
- Name
- Skript / Batch / Executable
- Parameter
- Beschreibung

Es ist zu evaluieren, wie die Auftragsdefinition abgespeichert wird. Pro Auftragsdefinition wird ein separates Verzeichnis angelegt. Zusätzlich benötigte Daten wie Skripte oder benötigte statische Dateien usw. werden ebenfalls hier abgelegt und mit jedem Auftrag neu an den Berechnungsserver ausgeliefert.

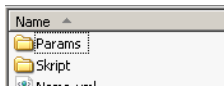


Abbildung 4: Beispiel einer möglichen Orderstruktur

### 3.3.3 Aufträge

Ein Auftrag wird aus einer Auftragsdefinition erstellt und beinhaltet alle benötigten Angaben, damit er an einen Berechnungsserver ausgeliefert und dort berechnet werden kann. Ein Auftrag ist in sich abgeschlossen. Ein Auftrag ist die kleinste mögliche Einheit.

Zusätzlich zu den Angaben aus der Auftragsdefinition beinhaltet ein Auftrag folgende Angaben

- Eindeutige ID
- Name
- Priorität
- Status (In Bearbeitung, Neu, Abgeschlossen, Fehler, Abgebrochen)
- Inputdaten
- Outputdaten

Es ist zu evaluieren, wie der eigentliche Auftrag abgespeichert wird. Pro Auftrag wird ein separates Verzeichnis angelegt. Neben einer Kopie der Auftragsdefinition werden die Inputdaten im jeweiligen Verzeichnis abgelegt. Outputdaten werden vom Berechnungsserver auch in diesem Verzeichnis gespeichert. Der Zugriff des Berechnungsservers auf die Daten ist zu evaluieren.

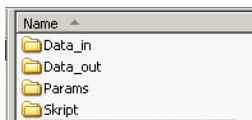


Abbildung 5: Beispiel einer möglichen Orderstruktur

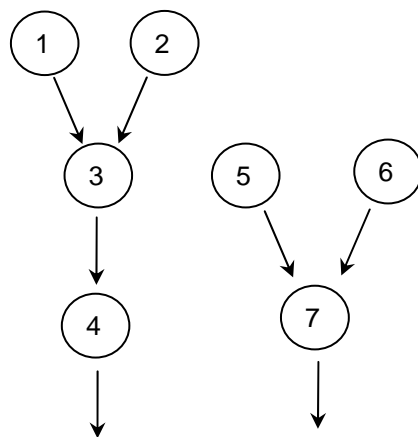
### 3.3.4 Sammelauftrag

Mehrere Aufträge werden in einem Sammelauftrag zusammengefasst. Die Reihenfolge der einzelnen Aufträge wird eingehalten. Die Aufträge können in Abhängigkeit eines anderen Auftrags sein. Ein einzelner Auftrag in einem Sammelauftrag kann max. zwei vorgängige Aufträge als Startbedingung haben. Der Sammelauftrag wird auf dem Auftragsserver verarbeitet und als einzelne Aufträge an die Berechnungsserver weitergeleitet.

Als Ergänzung werden dem Auftrag zusätzlich folgende Angaben hinzugefügt

- Eindeutige ID Sammelauftrag
- Name Sammelauftrag

Beispiel eines Sammelauftrags:



Auftrag 1, 2, 5 und 6 können direkt verteilt werden.

Auftrag 3 wird verteilt sobald Auftrag 1 und 2 abgeschlossen sind, Auftrag 7 wenn Auftrag 5 und 6 abgeschlossen sind.

Auftrag 4 wird nach Abschluss von Auftrag 3 verteilt.

Abbildung 6: Beispiel eines Sammelauftrags

### 3.3.5 Auslastungsreport

Der Auslastungsreport gibt einen Status der aktuellen Auslastung eines Berechnungsservers wieder. Er beinhaltet folgende Angaben:

- Allgemeine Infos
  - Maschinename
  - Betriebssystem
  - IP
- CPU
  - Anzahl Cores
  - Taktfrequenz
  - Auslastung Server
- Arbeitsspeicher
  - Maximal
  - Verfügbar
- Faktor (Optional)
  - Berechneter Performancefaktor

Aus Erfahrung lastet ein Auftrag einen Core zu 100% aus. Wenn nun ein Server mit 4 Cores zur Verfügung steht und dieser zu 25% ausgelastet ist, bedeutet dies, dass maximal weitere 3 Aufträge an diesen Server verteilt werden können.

Als Performancefaktor dient z.B. die Dauer für die Berechnung der Fakultät von 99'999.

## **3.4 Sicherheit**

### **3.4.1 Komponenten**

Für den Betrieb ist ein bereits geschütztes Umfeld vorgesehen. Damit werden nur Massnahmen für einen stabilen Betrieb benötigt, d.h. dass nicht aus Versehen Funktionen von aussen angestossen werden können. Die Sicherheitsvorkehrungen für die Dienste sind zu evaluieren.

### **3.4.2 Benutzer**

Es wird unterschieden zwischen Benutzern und Administratoren. Die Zugangsberechtigung wird in der Weboberfläche geprüft. Die Berechtigungen sind in Kapitel 3.1.4 beschrieben.

Es können mehrere Benutzer gleichzeitig eingeloggt sein. Es ist zu evaluieren, ob die Prüfung gegen das ActiveDirectory vorgenommen werden soll oder eine andere Prüfung verwendet wird.

## **4 Tests**

Ein Sammelauftrag, welcher in dieser Form im heutigen Umfeld eingesetzt wird, wird verwendet um die Funktion und die verbesserte Rechenzeit zu überprüfen.

Als Test dient die Berechnung von 16 EDGE-Layern. Die Rechenzeit für diesen Sammelauftrag beträgt aktuell 20h.



## 5 Zeitplan

	KW	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
<b>Termine</b>																												
Beginn Diplomarbeit (10. März)																												
Kickoff Meeting (27. März)																												
Abgabe Pflichtenheft (24. April)																												
Kurs Präsentationstechnik (6. Mai)																												
Bestätigung vom Pflichtenheft (8. Mai)																												
Abgabe Diplomarbeit (11. Sept)																												
<b>SOAPCALC</b>																												
Systemanalyse																												
Pflichtenheft erstellen																												
Prototyping																												
Grobdesign																												
Detailldesign																												
Implementation																												
Test																												
Dokumentation																												
Ferien																												
Milestones						1						2								3								4
		Inception						Elaboration						Construction I									Construction II					

### 5.1 Milestones

Milestone 1: Die Grobanalyse des Systems ist abgeschlossen, das Pflichtenheft mit den Anforderungen erstellt

Milestone 2: Die Prototypen des Systems funktionieren soweit, das die Kommunikation zwischen den Diensten funktioniert. Ein Pseudoauftrag kann an einen Berechnungsserver verteilt werden, dieser wird verarbeitet. Die Aktionen werden geloggt.

Milestone 3: Die Dienste Berechnungsserver und Logger sind abgeschlossen. Der Auftragsserver verwaltet Auftragsdefinitionen und Aufträge. Die Weboberfläche kann diese Funktionen nutzen.

Milestone 4: Projektabschluss

## 6 Anhang

### 6.1 Anforderungen

Anforderung Nr.	Beschreibung	Komponente	Priorität
10	Der Auftragsserver verwaltet die Aufträge.	Auftragsserver	1
10.1	Neue Aufträge können aus Auftragsdefinitionen erfasst werden	Auftragsserver	1
10.2	Bestehende Aufträge können verändert werden	Auftragsserver	2
10.3	Ein Auftrag wird abgespeichert	Auftragsserver	1
10.4	In einem Verzeichnis werden die benötigten Daten eines Auftrags abgelegt	Auftragsserver	1
10.5	Eine Auftragsübersicht kann abgefragt werden	Auftragsserver	1
10.6	Ein Auftrag kann gelöscht werden	Auftragsserver	2
11	Der Auftragsserver verwaltet die Auftragsdefinitionen	Auftragsserver	1
11.1	Neue Auftragsdefinitionen können erstellt werden	Auftragsserver	1
11.2	Bestehende Auftragsdefinitionen können verändert werden	Auftragsserver	2
11.3	Eine Auftragsdefinitionen wird abgespeichert	Auftragsserver	1
11.4	In einem Verzeichnis werden die benötigten Daten einer Auftragsdefinitionen abgelegt	Auftragsserver	1
11.5	Eine Auftragsdefinitionsübersicht kann abgefragt werden	Auftragsserver	1
11.6	Eine Auftragsdefinitionen kann gelöscht werden	Auftragsserver	2
12	Der Auftragsserver verwaltet die Sammelaufträge	Auftragsserver	1
12.1	Sammelaufträge können erstellt werden	Auftragsserver	1
12.2	Sammelaufträge können verändert werden	Auftragsserver	2
12.3	Ein Sammelauftrag kann gelöscht werden	Auftragsserver	2
13	Der Auftragsserver verteilt die Aufträge an die Berechnungsserver.	Auftragsserver	1
13.1	Wenn ein Berechnungsserver freie Kapazität hat, wird der Auftrag dem Berechnungsserver zugewiesen.	Auftragsserver	1
13.2	Wenn kein Berechnungsserver freie Kapazität hat, wird der Auftrag in eine Warteschlange gelegt.	Auftragsserver	1
13.3	Die Priorität eines Auftrags wird beim Verteilen berücksichtigt.	Auftragsserver	1
14	Aufträge können abgebrochen werden.	Auftragsserver	1
14.1	Die Verteilung eines Auftrags in der Warteschlange wird abgebrochen	Auftragsserver	1
14.2	Ein Auftrag in Bearbeitung wird auf dem Berechnungsserver abgebrochen.	Auftragsserver	1
14.3	Die Daten von abgebrochenen Aufträgen werden gelöscht.	Auftragsserver	1
15	Der Auftragsserver verwaltet die angemeldeten Berechnungsserver	Auftragsserver	1
15.1	Eine Übersicht der angemeldeten Berechnungsserver kann abgefragt werden	Auftragsserver	1
15.2	Der Auslastungsreport eines Berechnungsservers kann angefordert werden	Auftragsserver	1
15.3	Die Testroutine für einen Berechnungsserver kann gestartet werden	Auftragsserver	2
16	Der Auftragsserver stellt Methoden zur Verwaltung zur Verfügung	Auftragsserver	1

16.1	Die Konfiguration des Auftragsserver kann vorgenommen werden (Pfade, Prioritäten usw.)	Auftragsserver	1
16.2	Die Auswertung des Logs kann hier vorgenommen werden.	Auftragsserver	1
17	Es kann auch nach aktiven Berechnungsservern gesucht werden.	Auftragsserver	2
20	Dem Benutzer steht eine Weboberfläche zur Verfügung, um das System zu nutzen.	Weboberfläche	1
21	Mit der Weboberfläche kann man Aufträge verwalten	Weboberfläche	1
21	Die Weboberfläche kann mit einem Webbrowser angesprochen werden.	Weboberfläche	1
22	Eine Übersicht der Berechnungsserver und deren Auslastung kann angezeigt werden.	Weboberfläche	1
23	Mit der Weboberfläche kann man das System Administrieren	Weboberfläche	1
24	Eine Benutzerverwaltung erlaubt das Administrieren der Benutzer.	Weboberfläche	2
30	Ein Berechnungsserver verarbeitet die Aufträge.	Berechnungsserver	1
30.1	Der Berechnungsserver nimmt die Aufträge entgegen.	Berechnungsserver	1
30.2	Der Berechnungsserver startet die Aufträge in einem sep. Prozess.	Berechnungsserver	1
30.3	Die Prozess-ID des Prozesses, der den Auftrag ausführt, dem Auftragsserver zurückgegeben.	Berechnungsserver	1
30.4	Nach Abschluss des Auftrags wird eine Meldung an den Auftragsserver gesendet	Berechnungsserver	1
31	Ein Berechnungsserver kann sich am Auftragsserver anmelden	Berechnungsserver	2
32	Ein Berechnungsserver kann sich am Auftragsserver abmelden	Berechnungsserver	2
33	Ein Berechnungsserver kann einen Auftrag ablehnen, wenn z.B. ein benötigtes Zusatzprogramm wie MapInfo nicht installiert ist.	Berechnungsserver	3
34	Der aktuelle Auslastungsreport kann angefordert werden.	Berechnungsserver	1
35	Der Berechnungsserver kann einen Performancefaktor berechnen.	Berechnungsserver	3
40	Ein zentraler Logger steht allen Komponenten von SOAPCALC zur Verfügung	Logger	1
41	Der Logger nimmt einen Log Eintrag entgegen	Logger	1
42	Der Logger schreibt den Log Eintrag in die Datenbank	Logger	1
50	Mit dem Log Analyzer können die Einträge nach allen Informationen ausgewertet werden	Logger	1
60	Die Datenbank enthält Tabellen für den Logger	Datenbank	1
70	Als Kommunikationsprotokoll soll ein bestehendes offenes Protokoll verwendet werden (z.B. SOAP).	Kommunikation	1
71	Der Benutzer greift per http oder allenfalls https auf die Weboberfläche zu.	Kommunikation	1
80	Die Zugangsberechtigung wird in der Weboberfläche geprüft.	Sicherheit	2
81	Die Sicherheitsvorkehrungen für die Dienste sind zu evaluieren.	Sicherheit	2

---

# Anhang B: Quellenverzeichnis

## Literatur

Handbuch der .NET-Programmierung, Microsoft Press, Rolf Wenger  
ISBN-13: 978-3-86645-419-4

Microsoft Windows Communication Foundation Step by Step, Microsoft Press, John Sharp  
ISBN-13: 978-0-7356-2336-1

## Links

Allg. Information  
<http://www.wikipedia.ch>  
<http://www.w3schools.com>

Transaction Processing Performance Council  
<http://www.tpc.org/>

Visual Studio 2008 Update Infos  
<http://msdn.microsoft.com/de-de/vs2008/bb894691.aspx>

.NET Konfiguration  
[http://msdn.microsoft.com/en-us/library/2bc0cxhc\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/2bc0cxhc(VS.80).aspx)

Web Client Software Factory  
<http://www.codeplex.com/websf>  
<http://msdn.microsoft.com/en-us/library/bb264518.aspx>  
<http://blogs.msdn.com/msdnat/archive/2006/09/23/767816.aspx>

Web Service Software Factory  
<http://www.codeplex.com/servicefactory>  
<http://msdn2.microsoft.com/de-ch/magazine/cc163481.aspx>  
[http://msdn.microsoft.com/de-de/library/cc487895\(en-us\).aspx](http://msdn.microsoft.com/de-de/library/cc487895(en-us).aspx)  
<http://msdn.microsoft.com/en-us/library/cc304792.aspx>

Enterprise Library 4.0 für Visual Studio 2008  
<http://www.codeplex.com/entlib>

WCF Callback  
[http://www.c-sharpcorner.com/UploadFile/john\\_charles/CallbackoperationsinWindowsCommunicationFoundation06112007120551PM/CallbackoperationsinWindowsCommunicationFoundation.aspx](http://www.c-sharpcorner.com/UploadFile/john_charles/CallbackoperationsinWindowsCommunicationFoundation06112007120551PM/CallbackoperationsinWindowsCommunicationFoundation.aspx)  
[http://dotnetaddict.dotnetdevelopersjournal.com/wcf\\_alarmclock.htm](http://dotnetaddict.dotnetdevelopersjournal.com/wcf_alarmclock.htm)

101 LINQ Examples  
<http://msdn.microsoft.com/en-us/vcsharp/aa336746.aspx>

Web Controls  
<http://www.codeplex.com/VelodocXP>  
<http://www.codeplex.com/SharpPieces>

Bibliotheken  
<http://www.codeplex.com/NGenerics>  
<http://www.codeplex.com/DotNetZip>

## Eingesetzte Tools

SandCastle

<http://www.microsoft.com/downloads/details.aspx?FamilyId=E82EA71D-DA89-42EE-A715-696E3A4873B2&displaylang=en>

SandCastle Help File Builder

<http://www.codeplex.com/SHFB>

Sandcastle Styles Project

<http://www.codeplex.com/SandcastleStyles>

Visual Studio SDK

<http://msdn.microsoft.com/en-us/library/bb166441.aspx>

GhostDoc

<http://www.roland-weigelt.de/ghostdoc/>

## Anhang C: Abbildungsverzeichnis

Abbildung 1: Anwendungsbereich von SoapCalc .....	1
Abbildung 2: Mindmap SoapCalc .....	4
Abbildung 3: System - Schema .....	5
Abbildung 4: Zur Verfügung stehende Hardware .....	6
Abbildung 5: Beispiel eines Sammelauftrags .....	11
Abbildung 6: Tabellen der SoapCalc Datenbank .....	12
Abbildung 7: Übersicht File Handling .....	13
Abbildung 8: Ordnerstruktur des File Stores .....	13
Abbildung 9: Anwendungsfälle Auftragsserver.....	15
Abbildung 10: Anwendungsfälle Berechnungsserver.....	15
Abbildung 11: Anwendungsfall Log Server.....	16
Abbildung 12: Anwendungsfälle Log Analyzer.....	16
Abbildung 13: Anwendungsfälle Weboberfläche, Aufträge verwalten.....	16
Abbildung 14: Anwendungsfälle Weboberfläche, Berechnungsserver .....	17
Abbildung 15: Anwendungsfälle Weboberfläche, Administrationsbereich .....	17
Abbildung 16: Neuer Job .....	27
Abbildung 17: Ablauf Auftrag vorbereiten.....	28
Abbildung 18: Ablauf Auftrag verteilen .....	29
Abbildung 19: Auswahl der Berechnungsserver .....	30
Abbildung 20: Abmeldung Berechnungsserver vom Auftragsserver.....	31
Abbildung 21: Ablauf Neuer Log Eintrag .....	32
Abbildung 22: Log Eintrag in DB speichern.....	32
Abbildung 23: Anwendungsbereich der WCSF .....	33
Abbildung 24: Die in der WCSF vorhandene Funktionalität .....	33
Abbildung 25: Auftragsserver, Struktur der Solution .....	34
Abbildung 26: Schema Auftragsserver .....	35
Abbildung 27: Klassendiagramm Auftragsserver .....	36
Abbildung 28: SoapMain Service Contract.....	37
Abbildung 29: SoapMain Service Contract, Teil der Auftragsverarbeitung.....	38
Abbildung 30: SoapMain Service Contract, Teil der Auftragsdefinitionen.....	39
Abbildung 31: SoapMain Service Contract, Teil der Sammelaufträge .....	40
Abbildung 32: SoapMain Service Contract, Teil der Berechnungsserver .....	41
Abbildung 33: SoapMain Service Contract, Teil der Einstellungen .....	41
Abbildung 34: SoapMain Data Contract .....	42
Abbildung 35: SoapMain Fault Contracts .....	42
Abbildung 36: SoapMain Host Model .....	42
Abbildung 37: Die Klasse SoapMainDBController .....	43
Abbildung 38: Tabelle JobsDB .....	44
Abbildung 39: Tabelle CalcServerDB .....	44
Abbildung 40: Tabelle DefinitionsDB .....	44
Abbildung 41: Tabelle DefinitionParamsDB .....	44
Abbildung 42: Tabelle CollectiveDB .....	44
Abbildung 43: Tabelle CollectiveJobsDB.....	44
Abbildung 44: Die Klasse SoapMainCalcServer und die Proxyklasse.....	45
Abbildung 45: Die Klasse SoapMainLogger und die Proxyklasse .....	45
Abbildung 46: Die Klasse SyncEvents .....	46
Abbildung 47: Die SoapMain Settings .....	46
Abbildung 48: Threads und Prozesse des Berechnungsservers .....	47
Abbildung 49: SoapCalc Service Contract .....	48
Abbildung 50: SoapCalc Data- und Fault Contract .....	48
Abbildung 51: Die Klasse SoapCalcLogger und die Proxyklasse .....	49
Abbildung 52: Schema Log Server.....	50
Abbildung 53: SoapLogger Service Contract .....	50
Abbildung 54: SoapLogger Data- und Fault Contract .....	51
Abbildung 55: Definition Tabelle LoggerDB .....	51

---

Abbildung 56: SoapLogAnalyzer Service Contract .....	52
Abbildung 57: MVP Pattern .....	53
Abbildung 58: MVP Klassendiagramm .....	54
Abbildung 59: Solution der Weboberfläche .....	55
Abbildung 60: View mit ObjectContainerDataSource .....	56
Abbildung 61: File Browser Control .....	56
Abbildung 62: Beispiel einer InfoBox .....	56
Abbildung 63: Popup Template als MessageBox .....	56
Abbildung 64: Menu des Moduls SoapCalc .....	57
Abbildung 65: User Control des Sammelauftrags .....	57
Abbildung 66: Menu des Moduls Administration .....	58
Abbildung 67: Log Analyzer .....	58
Abbildung 68: Nachrichtenfolge Weboberfläche - Auftragsserver .....	60
Abbildung 69: Nachrichtenfolge Auftragsserver - Logger .....	61
Abbildung 70: Nachrichtenfolge Auftragsserver – Berechnungsserver für die Ausführung eines Auftrags ....	62
Abbildung 71: Initialisierung der Objekte .....	63
Abbildung 72: Service Contract vom Prototyp Logger .....	65
Abbildung 73: Data Contract vom Prototyp Logger .....	65
Abbildung 74: Host Explorer .....	66
Abbildung 75: Struktur des WCSF Prototyps .....	67
Abbildung 76: Tabelle im LINQ to SQL DataContext .....	67

## Anhang D: Informationen zum Deployment

Um die WCF Dienste auf einem Server zu betreiben, sind folgende Konfigurationen vorzunehmen:  
Nach der Installation des IIS muss die .NET Version 3.5 installiert werden.

Wenn diese Reihenfolge nicht eingehalten wurde, muss ASP.NET nachträglich registriert werden

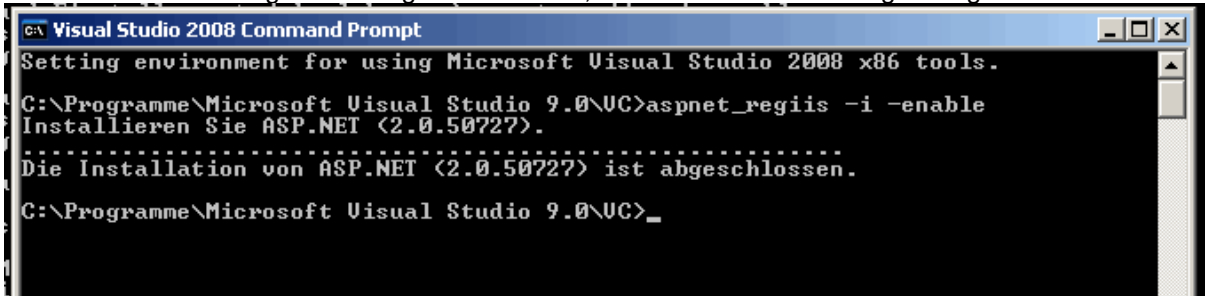


Abbildung D1: Registrierung von ASP.NET

Anschließend richtet man ein virtuelles Verzeichnis auf dem IIS für jeden Dienst auf dem Server ein.

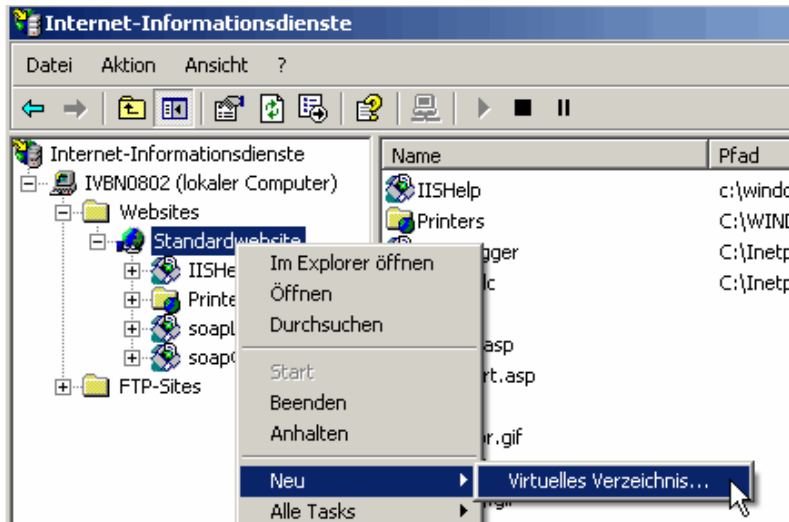


Abbildung D2: Einrichten eines virtuellen Verzeichnisses auf dem IIS

Danach wird die ASP.NET Version 2.0.50727 ausgewählt

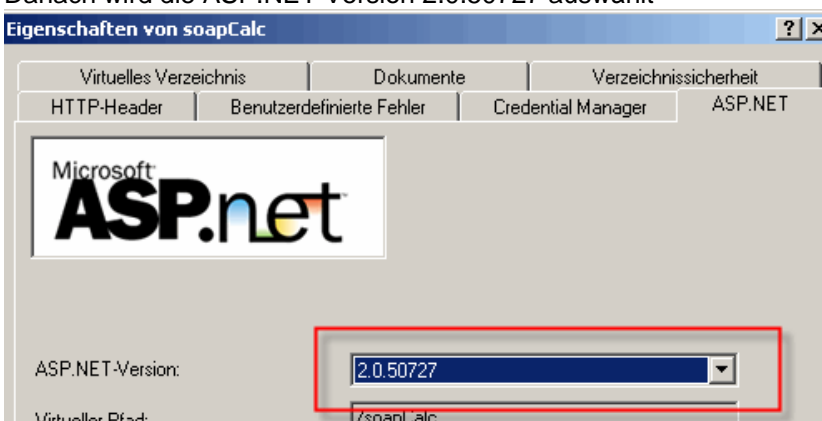


Abbildung D3: Auswahl der ASP.NET Version

Die mit dem Visual Studio gebildeten Files können nun in den entsprechenden Order kopiert werden.  
Benötigte Verzeichnisse für den Filestore müssen manuell angelegt werden. Als Test dient ein Aufruf des WSDL des Dienstes.



Auf dem Auftrags- bzw. Loggserver muss die Installation von SQL Express vorgenommen werden. Anschliessend erteilt man dem Benutzer, welcher im Application Pool des IIS die Dienste startet, die Berechtigungen für den Datenbankzugriff:

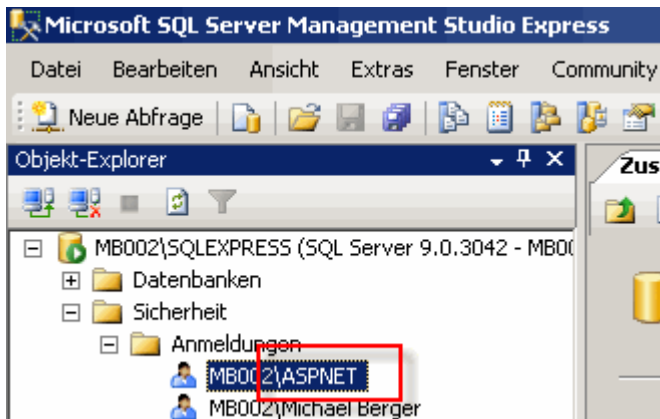


Abbildung D4: Benutzer ASPNET hinzufügen

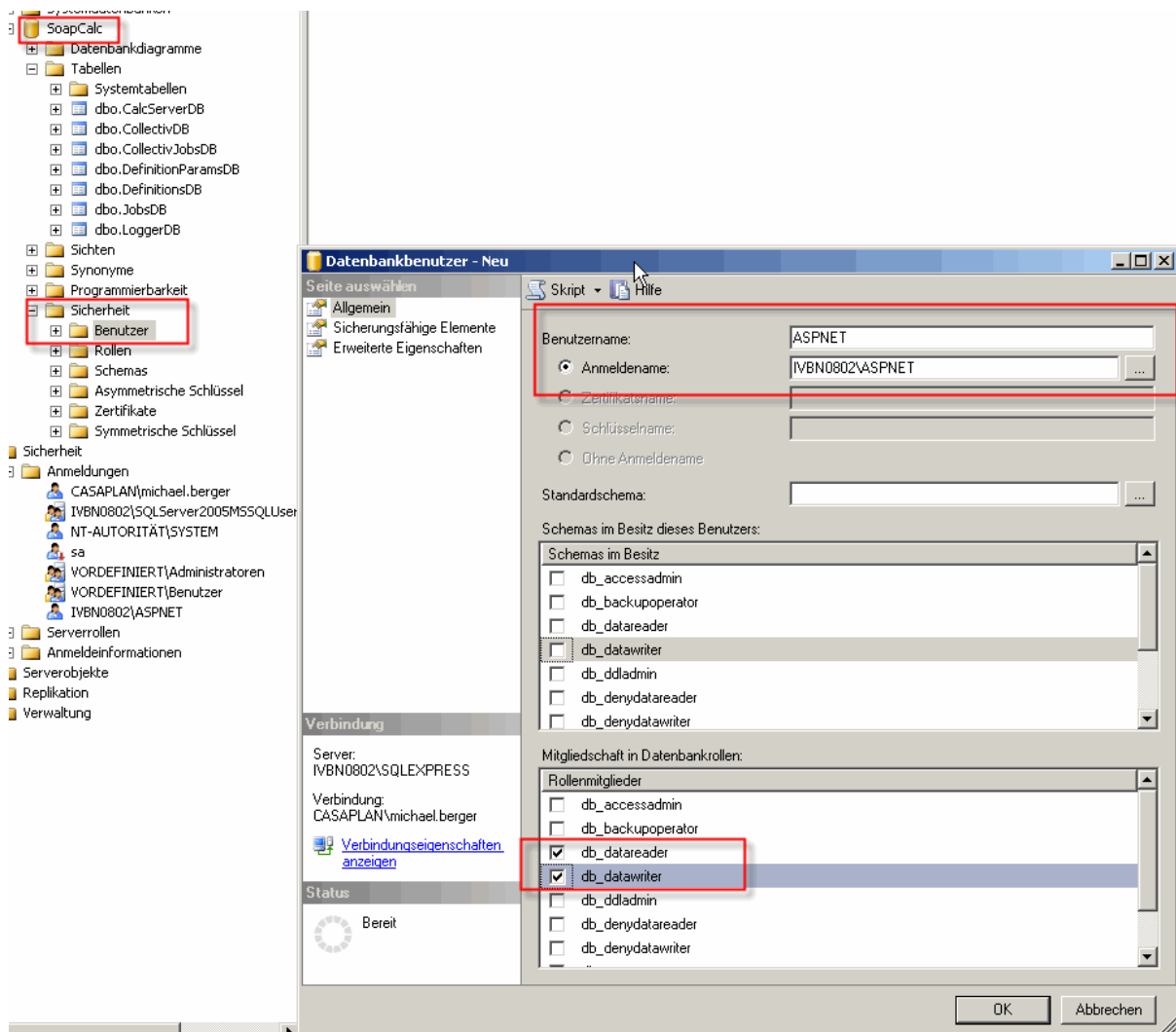


Abbildung D5: Rechte für Benutzer auf Datenbank geben

## Anhang E: Zugriff auf den Auftragsserver

### Web Zugriff

Die Weboberfläche ist unter folgender URL verfügbar:  
<http://212.60.50.178/soapWeb/>

Zwei Benutzer wurden eingerichtet: SoapUser und SoapAdmin.  
Beide verwenden das Passwort H@ber\$ack21!

### RDP

Gestartet wird die Remotedesktopverbindung auf der Konsole mit mstsc. Mit der Angabe der IP kann man sich auf den Auftragsserver verbinden.



Abbildung E1: Remotedesktopverbindung

Als Zugangsdaten verwendet man:



Abbildung E2: Zugangsdaten

### FTP

Um eine FTP Verbindung aufzubauen benötigt man folgende Zugangsdaten:

Server: 212.60.50.178  
User: bergadmin  
Passwort: H@ber\$ack21!