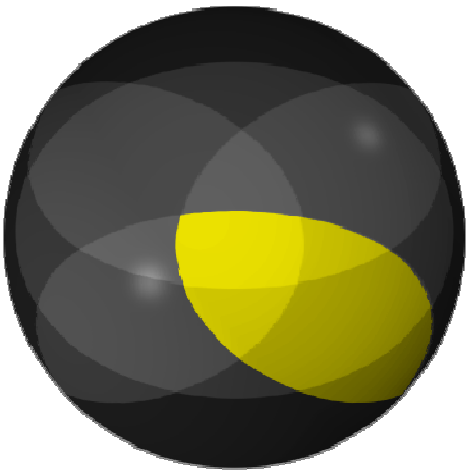
Christoph Rebsamen / Gieri Kohler 18.06.2010

Technischer Bericht

Version 1.0



Project Manager	Christoph Rebsamen
	Gieri Kohler
Betreuer	Prof. Beat Stettler
	Philipp Beck
Experte	Hansruedi Hänni, Luware AG
Industriepartner	Luware AG, Pfäffkon SZ

Dokumentenverwaltung

Dokumenthistorie

Version	Status	Datum	Verantwortlicher	Änderungsgrund
1.0	In process	09.03.2010	gk	Dokument erstellt
1.1	In process	04.04.2010	gk	Konzepte DW
1.2	In process	07.04.2010	gk	Technologien SSIS, SSAS, SSRS
1.3	In process	20.04.2010	cr	Chart Spezifikation
1.4	In process	20.04.2010	cr, gk	Anforderungsspezifikation
1.5	In process	30.04.2010	gk	Analyse bestehendes System
1.6	In process	01.05.2010	cr	Technologieanalyse
1.8	In process	07.05.2010	cr, gk	Architektur und Design
1.9	In process	08.05.2010	gk	Implementierung LUBI
1.10	In process	01.06.2010	cr	Implementierung LUBIWebFrontend
1.11	In process	15.06.2010	gk, cr	Management Summary
1.11.1	Finished	16.06.2010	gk, cr	Korrekturen

Review

Folgende Personen haben das Dokument gelesen und geprüft:

18.05.2010 / gk

16.06.2010 / gk

16.06.2010 / cr

Inhaltsverzeichnis

1 Einführung

1.1 Zweck

Dieses Dokument beschreibt die an der Hochschule für Technik Rapperswil (HSR) entwickelte Bachelorarbeit LUBI – Lean Unified Business Intelligence und deren Ergebnisse.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments beschränkt sich auf die Bachelorarbeit LUBI, welche im Frühlingssemester 2010, vom 22.02.2010 bis am 18.06.2010, durchgeführt wurde.

1.3 Übersicht

Im folgenden Kapitel findet sich das Management Summary, welches sich an alle Leser richtet. Die folgenden Kapitel richten sich an technisch versierte Leser, welche gegebenenfalls Erfahrungen auf dem Gebiet der Business Intelligence mitbringen. Kapitel 3 beschreibt die Aufgabenstellung der Bachelorarbeit. In Kapitel 4 sind die ausgearbeiteten Anforderungen in Form einer Anforderungsspezifikation zu finden. Kapitel 5 beschreibt die allgemein verwendeten Konzepte, die beim Design eines Data Warehouses verwendet werden. Die Analyse der verwendeten Technologien beschreibt das Kapitel 6. Die Analyse des bestehenden Systems LUCS, insbesondere der Live Datenbank, wird in Kapitel 7 dokumentiert. Kapitel 8 und 9 beschreiben die Architektur von LUBI und dem LUBIWebFrontend. Schlussendlich gehen Kapitel 10 und 11 auf interessante Punkte der Umsetzung und Implementierung der beiden Projektteile ein. Kapitel 12 gibt einen kurzen Ausblick in die Zukunft von LUBI. Schlussendlich findet sich in Kapitel 13 das Glossar der Arbeit.

2 Management Summary

2.1 Ausgangslage

Das Institute for Networked Solutions (INS) der Hochschule für Technik Rapperswil entwickelt für die Luware AG aufbauend auf eine Bachelor-Arbeit aus dem vergangenen Jahr ein Contact-Center-System. Es basiert auf dem Microsoft Office Communicaton Server (OCS) und verbindet verschiedenste Kommunikationsmedien wie Instant Messaging, Sprachanrufe oder Desktop-Sharing in einer einheitlichen Anwendung. Essentieller, aber komplexer Bauteil einer Contact-Center Lösung ist das Reporting-System. Elektronische Reporting-Systeme sind Teil des Business Intelligence (BI)-Gebietes, welches sich mit den Prozessen und Verfahren der Analyse von Geschäftsdaten in elektronischer Form beschäftigt.

Aufgabe dieser Bachelor-Arbeit ist das Entwerfen eines Konzepts für das Reporting-System und die Umsetzung desselbigen in einer Beispiel-Applikation. Die anfallenden Geschäftsdaten sollen erfasst, analysiert und anschliessend den verschiedenen Benutzerrollen in individueller Form präsentiert werden.

2.2 Vorgehen/Technologien

Nach einer eingehenden Analyse des Funktionsumfangs und der Möglichkeiten der zu verwendenden BI-Produkte von Microsoft und einer vertieften Abklärung der Anforderungen wurde ein Schema für die Ablage der Daten in einem Data Warehouse (DW) entwickelt. Ein Data-Warehouse soll die Speicherung grosser Datenmengen über lange Zeiträume und optimiert für die Weiterverarbeitung (z.B. Reporting) erlauben.

Die Integration der relevanten Daten aus der Live Datenbank wurde mittels der Integration Services in das Data Warehouse überführt. Die Daten werden anschliessend für die Analyse durch die Analysis Services in einen OLAP-Cube übernommen. Damit werden die Daten in ein mehrdimensionales Schema gespeichert. Vorteil dieser Darstellung der Daten ist die gleiche Weise, mit welcher auf verschiedene Aspekte (Dimensionen) der Daten zugegriffen wird. Um die Daten anschliessend für den Benutzer zu visualisieren, kommen die SQL Server Reporting Services für vorkonfektionierte, vorwiegend tabellarische Reports zum Einsatz. Als Frontend für den Gelegenheitsbenutzer wurde eine Silverlight-Webapplikation, welche auf den OLAP-Cube zugreift, entwickelt.

2.3 Ergebnisse

Die Bachelorarbeit befasste sich intensiv mit den Bereichen Data-Warehousing, Business Intelligence und Webapplikation-Entwicklung. Das zu Beginn erstellte Konzept wurde durch den entwickelten Prototyp verifiziert. Die erfassten Daten aus der Live Datenbank der Contact-Center-Lösung werden den unterschiedlichen Benutzerrollen Unternehmensleiter, Kundendienstleiter, Teamleiter und Kundenberater in ansprechender Form präsentiert. Um den Bedürfnissen aller Benutzerrollen gerecht zu werden, ist das Endprodukt in der Lage, unterschiedliche Reports zu generieren. Einerseits werden die Reporting Services 2008 von Microsoft verwendet, um tabellarische sowie grafische Reports anzubieten, welche in unterschiedlichen Formaten (beispielsweise PDF) in regelmässigen Abständen per E-Mail verschickt oder auf einen File Share kopiert werden können. Ebenfalls möglich ist das Betrachten der Reports über einen Webbrowser, wobei im Gegensatz zu einem PDF eine bessere Dynamik (Auswahl der anzuzeigenden Daten durch den Benutzer) erreicht werden kann. Andererseits wird ein Silverlight Web-Frontend angeboten, um grafisch anspruchsvolle Reports zu erzeugen, die über einen Webbrowser abgerufen werden können.

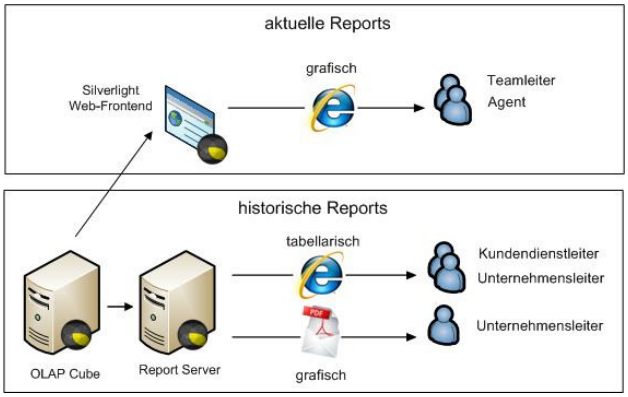


Abbildung 1: Benutzerrollen und Reportarten

Aufgrund der unterschiedlichen Bedürfnisse wird weiter unterschieden zwischen aktuellen Reports, die Daten der letzten Wochen beinhalten und historischen Reports, die weiter zurückliegende Daten (mehrere Jahre) berücksichtigen.

Die SQL Datenbank des LUCSM (Lean Unified Customer Service Manager) stellt die Schnittstelle der Bachelorarbeit LUBI zum Gesamtsystem her, in dem die Daten aus dieser SQL Datenbank gelesen, aufbereitet und ausgewertet werden:

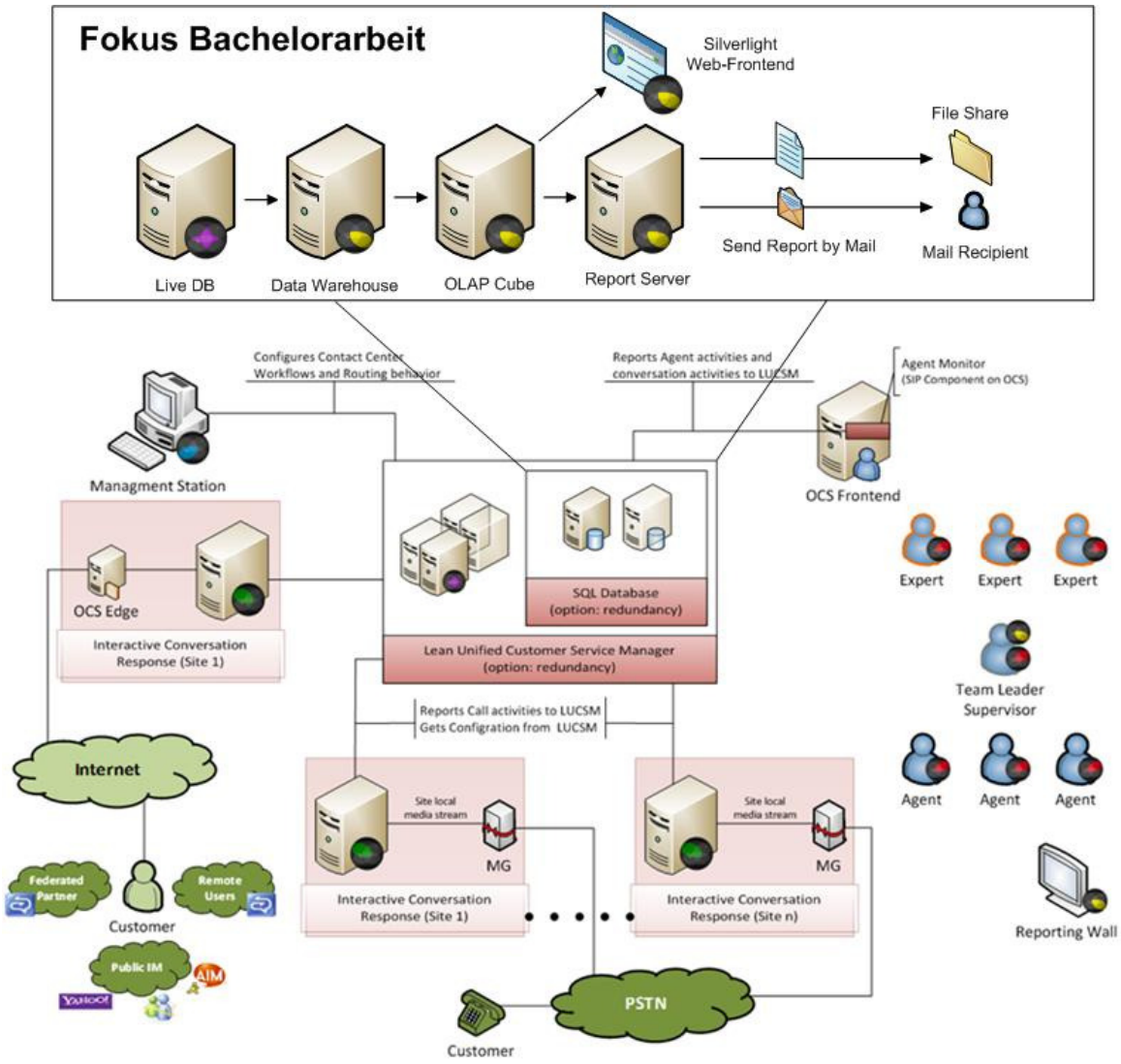


Abbildung 2: Systemübersicht

3 Aufgabenstellung

Die IP-Telefonie löst in vielen Firmen zunehmend die bewährte klassische Telefonie mit einem separaten Telefonnetz ab. Durch den Einsatz von Voice over IP (VoIP) Lösungen kann eine Konvergenz von Daten- und Sprachnetzwerk geschaffen werden. Der Softwarehersteller Microsoft vertreibt seit November 2007 das Produkt Office Communication Server (OCS), welches verschiedenste Kommunikationsmedien wie Instant Messaging, Sprachanrufe oder E-Mail in eine einheitliche Anwendungsumgebung integriert.
Der Industriepartner ist aktuell in der Entwicklung von einem neuen auf Microsoft Technologien basierenden Contact Centers. Ein sehr wichtiges und komplexes Bauteil eines jeden Contact Centers ist das Reporting-System.
Ziel dieser BA ist es, ein solches Reporting-System zu entwickeln. Dabei müssen historische Reports über einen Zeitraum von 5 Jahren dynamisch generiert werden können. Zusätzlich ist es wichtig, bei Schwellwertüberschreitungen vordefinierte Kontakte über die aktuellen Kommunikationsmedien wie E-Mail oder SMS zu informieren.
Zur BA gehört die Aufnahme aller Anforderungen, die Überführung der Daten aus der produktiven Datenbank in ein Datawarehouse, die Aufbereitung der Daten sowie die Erstellung von Reports. Die Reports sollen dem Endanwender in einem ansprechenden UI präsentiert werden. Für die Umsetzung müssen die Business Intelligence Technologien von Microsoft eingesetzt werden. Abgrenzung: Das Erstellen der produktiven Datenbank ist nicht Bestandteil der Bachelor Arbeit. Eine produktive Datenbank steht zur Verfügung. Die automatische Generierung von Events bei Überschreitung gewisser Schwellwerte ist nicht zwingender Inhalt der BA.

Unterschrift Betreuer: _____

4 Anforderungsspezifikation

4.1 Allgemeine Beschreibung

4.1.1 Produktfunktion

LUBI soll die Sammlung, Auswertung und Darstellung von Geschäftsdaten auf elektronischer Basis für die Contanct-Center-Lösung LUCS ermöglichen.

LUBI ermöglicht das Erstellen von statischen sowie dynamischen Reports der Geschäftszahlen eines Contact Centers über einen mehrjährigen Zeitraum. Diese Reports sollen als Grundlage für operative und strategische Entscheidungen im Hinblick auf die Unternehmensziele des Unternehmens, welches LUCS einsetzt, dienen.

Realisiert wird die systematische Analyse der Daten mithilfe der von Microsoft zur Verfügung gestellten BI-Werkzeuge aus der SQL-Server-Familie. Für die Darstellung der Geschäftszahlen zuhanden von Teamleitern und Agenten des Contact-Centers wird eine eigene Web-Applikation entwickelt. Diese ist auf die speziellen Bedürfnisse dieser Zielgruppe, auf welche im folgenden Kapitel spezifischer eingegangen wird, zugeschnitten.

4.1.2 Benutzercharakteristik

Für die Benutzercharakterisierung wurde mit dem Leiter einer Planning&Controlling-Abteilung einer Firma mit mehreren Contact-Centern ein Interview geführt. Als Vorbereitung auf das Interview wurde ein Interview-Plan erstellt (*03 Dokumentation/01 Analyse/ Interviewplan_Contact_Center_Reportingsystem.docx*). Die vor dem Interview entwickelten hypothetischen Personas wurden während dem Interview verifiziert. Die verifizierten Personas finden sich in einem separaten Dokument (*03 Dokumentation\01 Analyse\Personas_Szenarios_Contact_Center_Reportingsystem.docx).* Diese Dokumente wurden im Verlauf der UInt2-Moduls in Zusammenarbeit mit Philipp Marugg und Matthias Good, zwei weiteren HSR-Studenten erstellt. Die aus diesem Interview entstandenen Benutzercharakteristiken werden im Folgenden beschrieben.

4.1.2.1 Oberes Management

Angehörige des oberen Managements erwarten fertig konfektionierte Berichte mit den Geschäftszahlen. Diese Geschäftszahlen müssen entweder in tabellarischer oder in grafischer Form vorliegen. Diese Berichte sollen in regelmässigen Abständen ausgeliefert werden. Das kann bedeuten, dass sie per E-Mail an interessierte Bezüger gesendet werden oder zur Archivierung im unternehmensweiten Filesystem abgelegt werden.

4.1.2.2 Mitarbeiter Planning & Controlling

Mitarbeiter von Planning&Controlling-Abteilungen und vergleichbare Mitarbeiter sind oftmals, salopp gesagt, "Excel-Power-User" und wünschen direkten Zugriff auf die Rohdaten der Geschäftszahlen. Sie stellen sich ihre, oftmals sehr spezifischen, Reports selber im Excel zusammen.

4.1.2.3 Kundendienstleiter

Der Kundendienstleiter benötigt für seine tägliche Arbeit immer dieselben Reports. Er schätzt es, wenn diese als Tabellen und allenfalls als Grafiken vorliegen. Benötigt er Daten für eine spezfische Aufgabe, wendet er sich in Regel an die Profis in der Planning & Controlling-Abteilung. Kundendienstleiter führen oft zielorientiert, d.h. sie geben den Untergebenen Ziele vor (z.B. „Durchschnittliche Gesprächsdauer sollte zwischen vier und sechs Minuten liegen“). Bis jetzt konnten Untergebene nur schlecht selbst kontrollieren, ob sie solche Zielvorgaben erfüllen oder nicht. Mit dem WebFrontend von LUBI gibt der Kundendienstleiter seinen Untergebenen nun ein Werkzeug in die Hand, mit dem sie sich auf einfache Art und Weise selber kontrollieren können.

4.1.2.4 Teamleiter

Der Teamleiter interessiert sich vorallem für die Geschäftszahlen, welche die Einhaltung der Vorgaben an seine Untergebenen aufzeigen. Da der Teamleiter durch andere Aufgaben genügend ausgelastet ist, beschränkt sich seine Zeit, die er für das Studium der Geschäftszahlen aufbringen kann, auf ein Minimum. Es ist ihm deshalb wichtig, möglichst auf einen Blick und grafisch dargestellt die relevanten Daten präsentiert zu bekommen.

4.1.2.5 Kundenberater

Für den Kundenberater gilt wie oben erwähnt das Bedürfnis, zu wissen, wo man steht. Auch er hat keine Zeit für ein langes Studium der Geschäftszahlen und möchte seine relevanten Informationen möglichst kompakt, grafisch ansprechend und verständlich präsentiert bekommen. Agenten fehlt zudem oft das betriebswirtschaftliche Verständnis der Geschäftszahlen, deshalb müssen diese auf einfache und verständliche Art erklärt werden.

4.2 Allgemeine Anforderungen

4.2.1 Auslieferung Reports

Reporte sollen im PDF-Format per E-Mail verschickt werden können. Auch das Archivieren im Filesystem erfolgt im PDF-Format.

4.2.2 WebFrontend

Es soll ein WebFrontend erstellt werden, um der Anforderung nach "Access from Everywhere" gerecht zu werden.[1]

4.2.3 Eingesetzte Technologien

Es wird nochmals betont, dass für die Umsetzung auf die BI-Tools des Microsoft SQL-Server gesetzt wird. Diese Umfassen die folgenden Komponenten:

- SQL-Server Integration Services (SSIS)
- SQL-Server Analysis Services (SSAS)
- SQL-Server Reporting Services (SSRS)

Für die Umsetzung des WebFrontends soll Microsoft Silverlight eingesetzt werden.

4.3 Funktionale Anforderungen

4.3.1 Aktoren

Primäre Aktoren sind der Unternehmensleiter, der Kundendienstleiter, der Teamleiter und der Kundenberater.

Aktor	Beschreibung
Unternehmensleiter	Mitglied des höheren Managements. Übt in der Regel im Unternehmen eine Kontrollfunktion aus. Je nach Aufgabenstellung für gewisse Bereiche, z.B. Qualitätssicherung, Personalmanagement, etc. zuständig. Basiert seine Entscheide unter anderem auf ihm zugetragenen Geschäftszahlen. Der Unternehmensleiter hat Interesse an längerfristigen Trends. In grösseren Firmen übernimmt teilweise die Abteilung Planning&Controlling gewisse Aufgaben dieser Personen. Mitglieder der Planning&Controlling-Abteilung werden ebenfalls der Rolle Unternehmensleiter zugerechnet, verfügen aber im Gegensatz zu diesem in der Regel über fundiertere Excel- und BI-Kenntnisse..
Kundendienstleiter	Leitet das Contact-Center der Firma. Ist verantwortlich für die Arbeit und das Wohlergehen der Mitarbeiter des Contact-Centers. Er ist für die Einsatzplanung, die Einhaltung der Qualitätsmassnahmen und für das Daily-Business des Contact-Centers verantwortlich. Er ist in der Regel eher an kurzfristigen Rückblicken interessiert. Es kann aber vorkommen, dass auch er an längerfristigen Trends interessiert ist.
Teamleiter	Führt in der Regel ein Team von mehreren Kundenberatern, welche die Konversationen für gewisse Service Entry Points führen. Er setzt gegenüber seinen Untergebenen die Vorgaben

[1] Setzt eine funktionierende Internetverbindung voraus. Je nach Unternehmens-Policy ist ein VPN-Tunnel in die Firma nötig.

	und Ziele der vorgesetzten Stellen durch. In Zusammenarbeit mit dem Teamleiter erstellt er die Einsatzpläne seiner Kundenberater. Er hat ähnliche Interessen an Geschäftszahlen wie der Kundendienstleister, nur ist seine Sicht auf die Daten auf sein Team beschränkt. Der Teamleiter ist normalerweise gleichzeitig ein Kundenberater, sprich auch er beantwortet Kunden- oder interne Anfragen.
Kundenberater	Führt die Beratungen per Telefon, Instant-Messaging, Desktop-Sharing und allen weiteren Möglichkeiten, die LUCS bietet. Ist in der Regel nicht wirklich an den Geschäftszahlen interessiert, ausser sie betreffen ihn persönlich. Kann aber dankbar sein, wenn ihm ein Werkzeug in die Hand gegeben wird, mit dem er das Erreichen der ihm gesteckten persönlichen Ziele überwachen kann.

Tabelle 1: Aktoren

4.3.2 Userstories

Die Userstories wurden vom Vertreter des Industriepartners, Hansruedi Hänni geschrieben.

4.3.2.1 Unternehmensleiter

Als Unternehmensleiter will ich eine Cockpit-Sicht auf die Kundeninteraktionen und wie wir mit ihnen umgehen, so dass ich mein Unternehmen steuern kann.

Als Unternehmensleiter interessieren mich Trends, das heisst Veränderungen:

- Veränderungen bei den Zugangsarten: PSTN, Public IM Connectivity (AOL, MSN, Yahoo!), Communicator Web Access, Federation, internal customer.
- Veränderungen in den gewählten Modalitäten
- Veränderungen in den Beratungsarten (mehr Konferenzen etc.)
- Veränderungen in den Beratungslängen
- Veränderungen in den Spitzenzeiten der Anfragen
- Verschiebungen bei den Anfragen pro Produkt
- Auswirkungen einer neuen Conversation Distribution Policy (hat sich die Investition in LUCS gelohnt)

Ich möchte Vergleiche, z.B. zur selben Periode des Vorjahres ziehen können.

Anmerkung der Autoren:

Die Zugangsarten sind zurzeit nicht in den Daten in der Live-Datenbank verfügbar. Deshalb ist dieser Punkt der Userstory nicht erfüllt.

Für die Veränderungen in den Beratungsarten liegt kein Standart-Report vor, der diese Veränderungen zeigt. Die Daten sind jedoch im Cube vorhanden und können z.B. über Microsoft Excel abgefragt werden.

4.3.2.2 Kundendienstleiter

Als Kundendienstleiter will ich eine Cockpit-Sicht auf die Kundeninteraktionen meines Bereichs und wie wir mit ihnen umgehen, so dass ich meinen Bereich planen und steuern kann.

Als Kundendienstleiter interessieren mich ähnliche Trends wie den Unternehmensleiter, jedoch bezogen auf meine Geschäftseinheit. Zum Beispiel Verschiebungen bei den Anfragen pro Service.

Mich interessiert aber auch der operative Betrieb:

- Wie viele Anfragen konnten im primären Vertriebskreis bearbeitet werden, wie viele erst im sekundären oder tertiären?
- Wie viele Anfragen aus anderen Geschäftseinheiten wurden durch unser Business Unit behandelt?
- Wie viele Anfragen an unsere Geschäftseinheiten wurden durch andere Business Unit behandelt?

Mich interessieren natürlich auch die konventionellen Contact Center Kennzahlen wie Servicelevel, verlorene Anrufe, durchschnittliche Gesprächsdauer etc.

Anmerkung der Autoren:

Da sich Service Entry Points im jetzigen Live-Datenbank-Modell nicht einer Organisationseinheit zuordnen lassen, können auch keine Aussagen über die Anzahl Anfragen, welche von anderen Geschäftseinheiten beantwortet wurden oder solche, welche die eigene Geschäftseinheit für Andere beantwortet hat, gemacht werden.

4.3.2.3 Teamleiter

Als Teamleiter will ich die historischen Kennzahlen der Services meines Teams sehen, so dass ich mittel- und langfristig planen kann.

4.3.2.4 Kundenberater

Als Kundenberater will ich die jeweilige Auslastung der Services ,welche ich primär bediene, sehen, so dass ich entsprechend agieren kann.

4.3.3 Reportarten

Reporte können in vielfältiger Ausführung und Ausprägung erstellt werden. Folgende Tabelle soll die relevanten Report-Arten für LUBI beschreiben:

Report	Beschreibung	Zielgruppe
Tabellarischer, historischer Report	Die Geschäftszahlen werden in tabellarischer Form über Zeiträume dargestellt. Ist der Report interaktiv, können dem Benutzer Drill-Down-Aktionen auf dem Report angeboten werden, womit auf noch detailliertere Zahlen zugegriffen werden kann.	• Kundendienstleiter • (Unternehmensleiter)
Grafischer, historischer Report	Die Geschäftszahlen werden komprimiert für längere Zeiträume in Grafiken und Diagrammen dargestellt. Werden bevorzugt als PDF interessierten Personen zugestellt.	• Unternehmensleiter
Grafischer, kurzfristiger Report	Die Geschäftszahlen der vergangenen Wochen, allenfalls des vergangenen Monats werden in ansprechender Form (animierte Charts, interaktiv) dem Bezüger präsentiert. Hilfestellungen beim Interpretieren der Zahlen und Charts sind hier angebracht. In der Regel handelt es sich bei den Bezügern um Laien im Bezug auf Unternehmenszahlen.	• Teamleiter • Kundenberater
Live-Report (Nicht umgesetzt, der Vollständigkeit halber erwähnt)	Wiederspiegelt die aktuelle Situation/den aktuellen Zustand des Contact-Centers, sprich er basiert auf zeitnah ausgelesenen Daten aus der Live-Datenbank. Der Einsatz eines solchen Reports erfolgt oftmals in der Form einer Live-Wall, welche direkt im Contact-Center an die Wand projiziert wird.	• Teamleiter • Kundenberater

Tabelle 2: Reportarten

4.3.4 Chart Spezifikationen

Für die Aktoren Teamleiter und Kundenberater wurden die folgende Charts. Dies ist keine abschliessende Auflistung und stellt nur eine Standart-Menge an Charts dar.

4.3.4.1 Servicelevel-Chart

Stellt den Servicelevel für verschiedene Service Entry Points über einen Zeitraum dar.

	Wert
Art	Bar-Chart
X-Achse	Zeit
Y-Achse	Servicelevel in Prozent
Gruppierung	Zeitintervall
Spezielles	Durchschnitt pro Zeitintervall
Default	Anzeige des Servicelevels für alle betreuten ServiceEntryPoints des Teams für: • den vergangenen Tag • für die vergangenen drei Tage • für die vergangene Woche
Parametrisierungsmöglichkeiten	• Zeitraum • Service
Berechnung[2]	_____

Tabelle 3: Beschreibung Servicelevel-Chart

Beispiel:

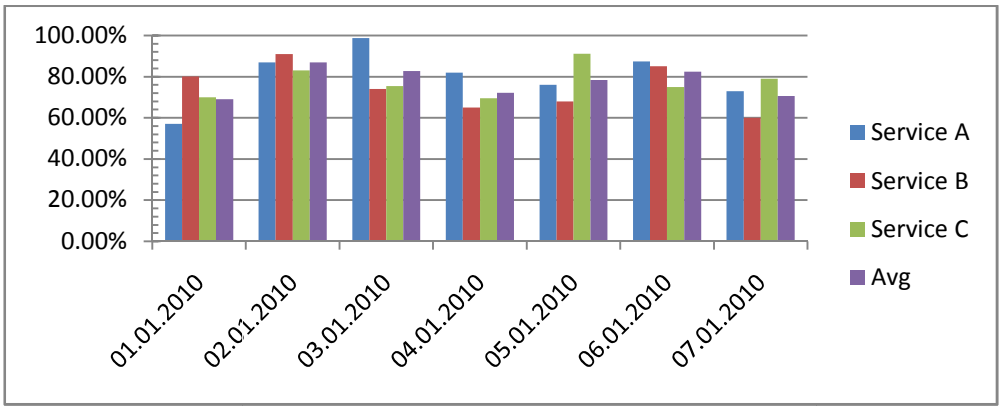


Abbildung 3: Chart Servicelevel

2

Handled<Param1: Handled Calls Within Time (Anzahl angenommene ServiceCalls innerhalb parametrisierbarer Zeitdauer)
IC: Incoming Calls (Anzahl eingehende ServiceCalls)
OoS: Out of Service Calls (Anzahl ServiceCalls ausserhalb der Servicezeiten)
LBQ: Lost Calls Before Queue (Anzahl ServiceCalls, die vor der Warteschlange abgebrochen wurden)
Abandoned<Param2: Abandoned Calls Within Time (Abgebrochene Calls innerhalb parametrisierbarer Zeitdauer)

4.3.4.2 Team Conversations-Chart

Zeigt die aufsummierten Konversationen des Teams über einen Zeitraum.

	Wert
Art	Horizontal-Stack-Bar-Chart
X-Achse	Absolute Zahl an Conversations
Y-Achse	Zeit
Default	Anzeige der Anrufe für alle ServiceEntryPoints des Teams aufgeschlüsselt nach "Beendigungszustand" für: • den vergangenen Tag • für die vergangenen drei Tage • für die vergangene Woche
Parametrisierungsmöglichkeiten	• Zeitraum • Service

Tabelle 4: Beschreibung Team Conversations-Chart

Beispiel:

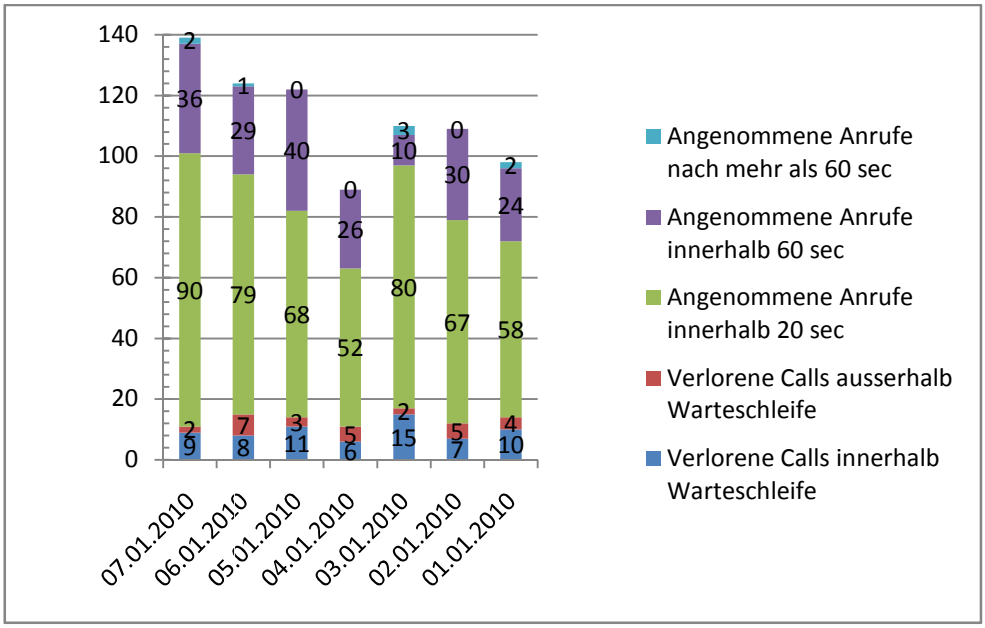


Abbildung 4: Chart Team Conversations

4.3.4.3 Conversations per Agent/ServiceEntryPoint-Chart

Zeigt die Anzahl Konversationen der einzelnen Kundenberater, aufgeteilt nach Service Entry Point.

	Wert
Art	Horizontal-Stack-Bar-Chart
X-Achse	Absolute Zahl an Conversations
Y-Achse	Kundenberater
Default	Anzeige der gemachten Beratungen pro Kundenberater aufgeschlüsselt nach ServiceEntryPoint über den festgelegten Zeitraum (vergangene Tag, vergangene 3 Tage, vergangene Woche)
Parametrisierungsmöglichkeiten	• Zeitraum • Service

Tabelle 5: Beschreibung Conversations per Agent/ServiceEntryPoint-Chart

Beispiel:

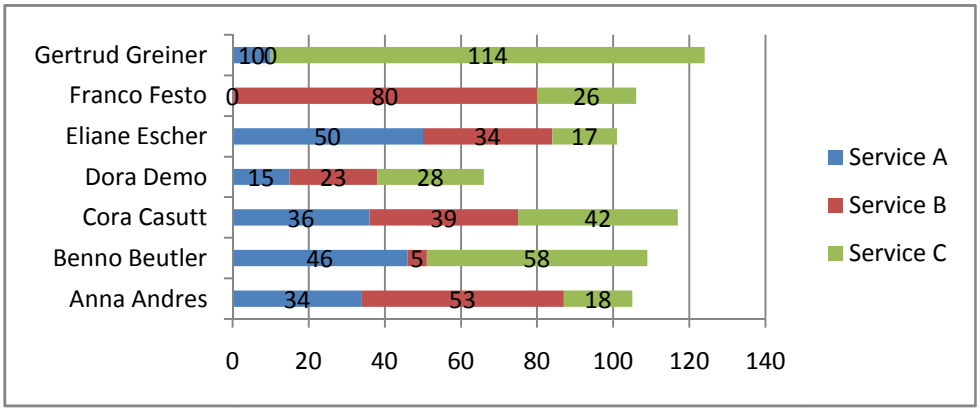


Abbildung 5: Chart Conversations per Agent / ServiceEntryPoint

4.3.4.4 Conversations per Agent-Chart

Zeigt die Anzahl Konversationen der einzelnen Kundenberater, kategorisiert nach dem Kriterium, wie schnell eine Konversation aufgenommen wurde.

	Wert
Art	Horizontal-Stack-Bar-Chart
X-Achse	Absolute Zahl an Conversations
Y-Achse	Kundenberater
Default	Anzeige der gehandelten Calls pro Kundenberater aufgeschlüsselt nacht Reaktionszeit (vergangene Tag, vergangene 3 Tage, vergangene Woche)
Parametrisierungsmöglichkeiten	• Zeitraum • Service

Tabelle 6: Beschreibung Conversations per Agent-Chart

Beispiel:

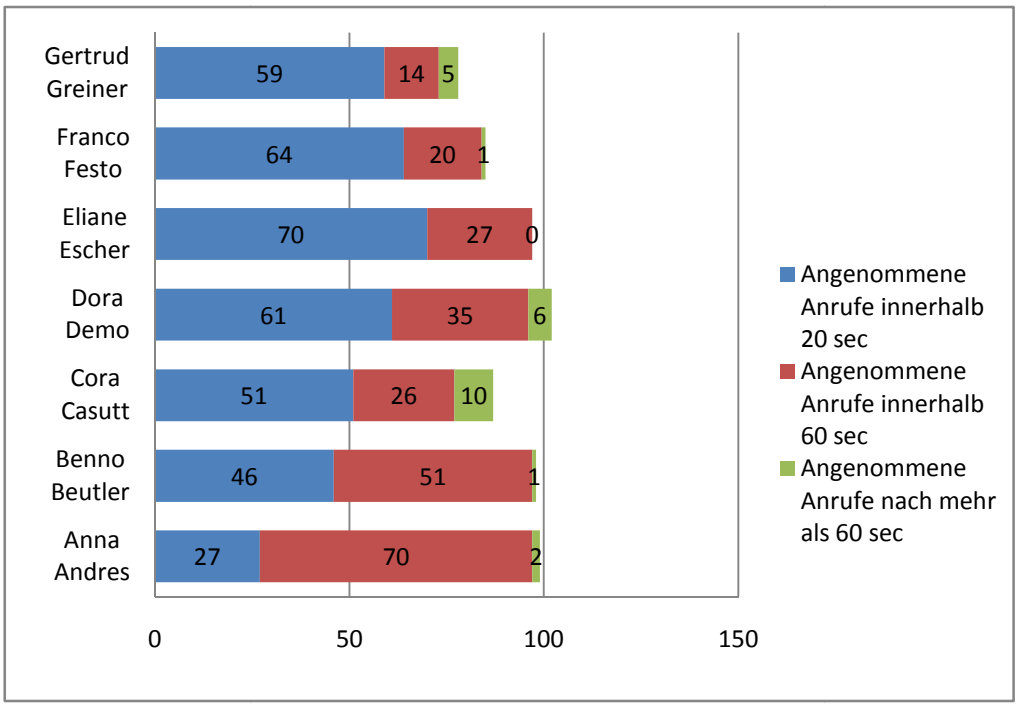


Abbildung 6: Chart Conversations per Agent

4.3.4.5 Initial Modalities-Chart

Stellt die benutzten Initial-Modalitäten der Konversationen für einen definierten Zeitraum als Kuchendiagramm dar.

	Wert
Art	Pie-Chart
Default	Verteilung der Modalitäten über den festgelegten Zeitraum (vergangene Tag, vergangene 3 Tage, vergangene Woche)
Parametrisierungsmöglichkeiten	• Zeitraum • Service

Tabelle 7: Beschreibung Initial Modalities-Chart

Beispiel:

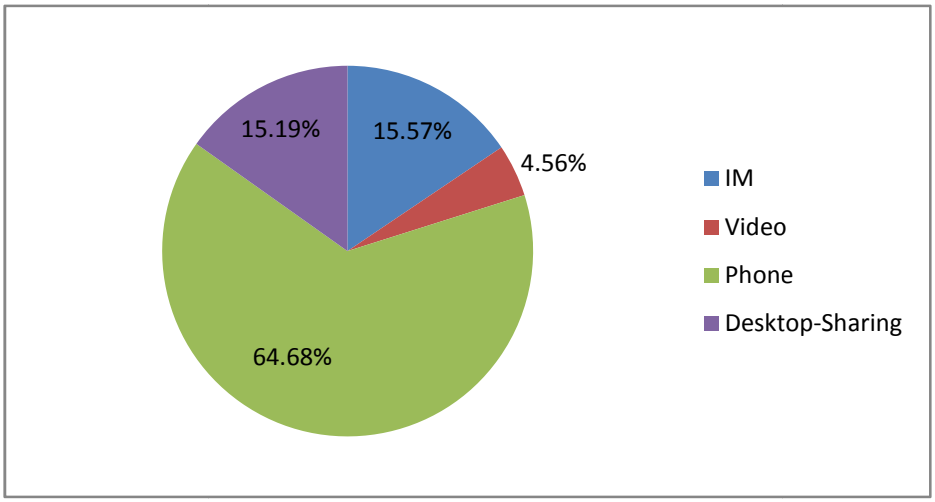


Abbildung 7: Chart Initial Modalities

4.3.4.6 Conversations Over Day-Chart

Stellt den Durchschnitt der geführten Konversationen eines Teams pro Stunde über einen Tag dar.

	Wert
Art	Spline-Chart
X-Achse	Zeit
Y-Achse	Absolute Zahl an Conversations (Durchschnitt)
Default	Verteilung der ankommenden Calls aufgespannt auf einen Tag während den Service-Zeiten über den gewählten Zeitraum gesehen. Besteht der Zeitraum aus mehreren Tagen wird der Durschnitt angezeigt
Parametrisierungsmöglichkeiten	• Zeitraum • Service

Tabelle 8: Beschreibung Conversations Over Day-Chart

Beispiel:

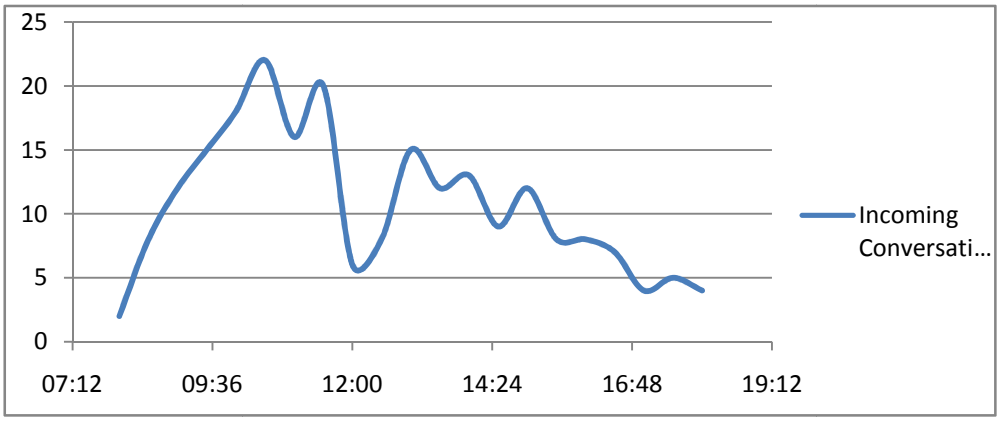


Abbildung 8: Chart Conversation Over Day

4.3.4.7 Queue Time-Chart

Stellt die Wartezeit des Kunden in der Warteschlange, aufgeteilt nach Service Entry Point für das gesamte Team dar.

	Wert
Art	Bar-Chart
X-Achse	Service
Y-Achse	Queue Time der Kunden (min:sec)
Gruppierung	ServiceEntryPoint
Default	Minimale, maximale und durchschnittliche Wartezeit des Kunden pro angebotenem ServiceEntryPoint über den eingestellten Zeitraum
Parametrisierungsmöglichkeiten	• Zeitraum • Service

Tabelle 9: Beschreibung Queue Time-Chart

Beispiel:

Abbildung 9: Queue Time

4.3.4.8 Talk Time Agent-Chart

Stellt die maximale und durchschnittliche Gesprächsdauer pro Kundenberater für einen Zeitraum dar.

	Wert
Art	Horizontal-Bar-Chart
X-Achse	Zeit
Y-Achse	Kundenberater
Gruppierung	Kundenberater
Default	Maximale und durchschnittliche Gesprächsdauer pro Kundenberater über den eingestellten Zeitraum
Parametrisierungsmöglichkeiten	• Zeitraum • Service

Tabelle 10: Beschreibung Talk Time Agent-Chart

Beispiel:

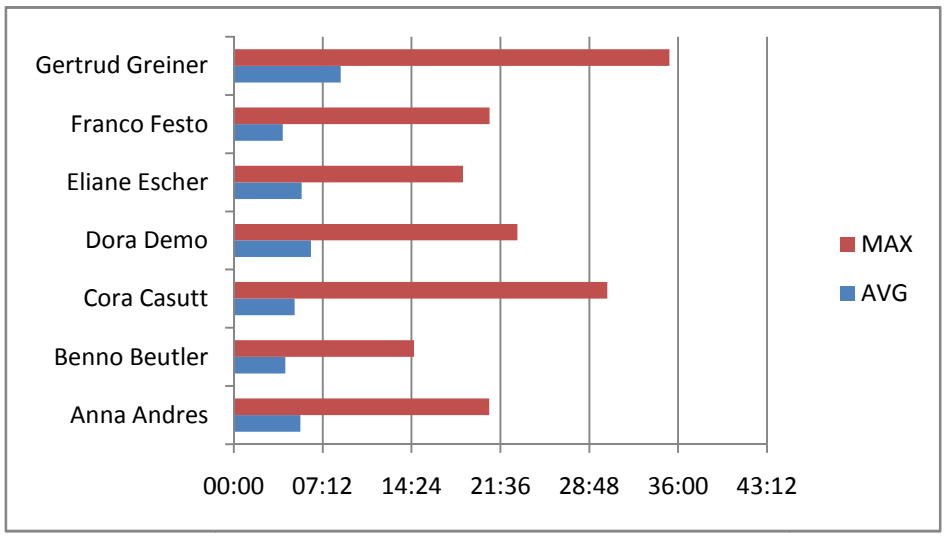


Abbildung 10: Chart Talk Time Agent

4.3.4.9 Acceptance Time Agent-Chart

Das Diagramm zeigt die durchschnittliche und maximale Zeit, die ein Kundenberater gebraucht hat, um eine Konversation anzunehmen.

	Wert
Art	Horizontal-Bar-Chart
X-Achse	Zeit
Y-Achse	Kundenberater
Gruppierung	Kundenberater
Default	Maximale und durchschnittliche Annahmezeit pro Kundenberater über den eingestellten Zeitraum
Parametrisierungsmöglichkeiten	• Zeitraum • Service

Tabelle 11: Beschreibung Acceptance Time Agent-Chart

Beispiel:

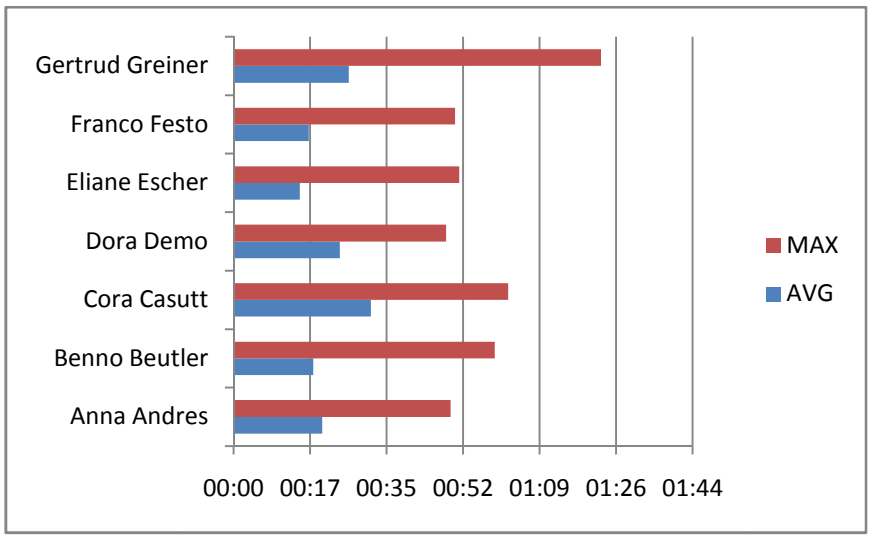


Abbildung 11: Chart Acceptance Time Agent

4.3.4.10 Non Service Conversations-Charts

Zeigt die Konversationen des Kundenberaters an, welche nicht über einen Service Entry Point reinkommen. Dies sind Konversationen innherhalb der Unternehmung (Internal), Direktkonversationen von aussen (Inbound) und Konversationen nach aussen (Outbound).

	Wert
Art	Horizontal-Bar-Chart
X-Achse	Absolute Zahl an Conversations
Y-Achse	Kundenberater
Gruppierung	Kundenberater
Default	Anzahl ausgehende und interne Konversationen pro Kundenberater während definierten Zeitraum
Parametrisierungsmöglichkeiten	• Zeitraum

Tabelle 12: Beschreibung Non Service Conversations-Chart

Beispiel:

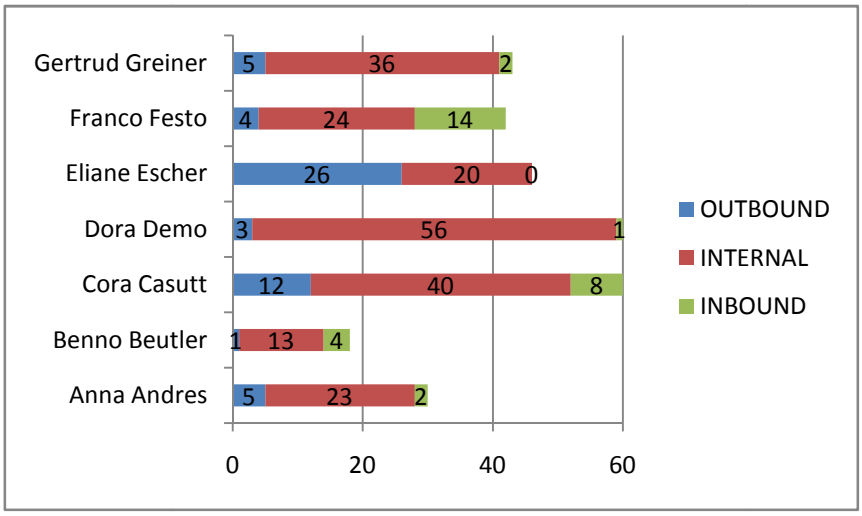


Abbildung 12: Chart Non Service Conversations

4.3.4.11 Talk Time Non Service Conversations-Chart

Zeigt die Gesprächs-Zeiten der Kundenberater für Konversationen die nicht über einen Service Entry Point gehen.

	Wert
Art	Horizontal-Bar-Chart
X-Achse	Gesprächszeit
Y-Achse	Kundenberater
Gruppierung	Kundenberater
Default	Durchschnittliche und maximale Dauer von ausgehenden und internen Konversationen pro Kundenberater während definierten Zeitraum
Parametrisierungsmöglichkeiten	• Zeitraum

Tabelle 13: Beschreibung Talkt Time Non Service Conversations-Chart

Beispiel:

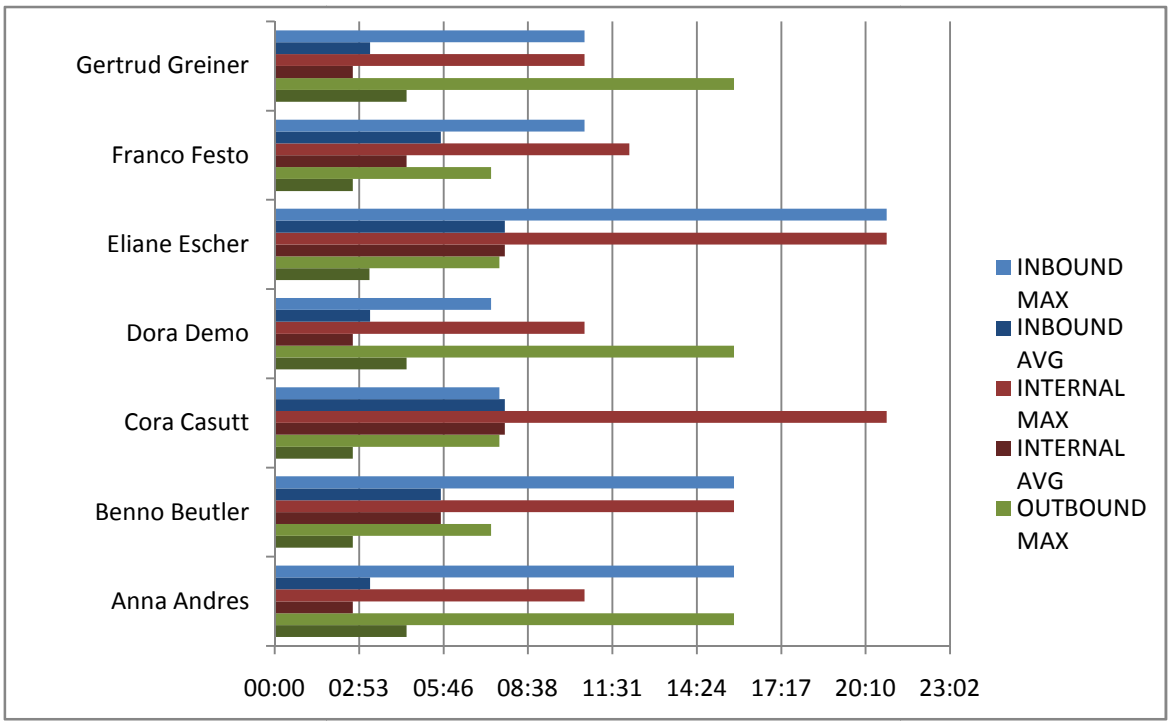


Abbildung 13: Chart Talk Time Non Service Conversations

4.4 Nichtfunktionale Anforderungen

4.4.1 Reifegrad

Es soll ein Proof-of-Concept erstellt werden. Das bedeutet, dass die funktionalen Anforderungen im Sinne einer Machbarkeitsstudie in einer Applikation umgesetzt werden sollen. Dabei werden Aspekte wie Einbettung in eine Authentifizierungs- und Authorisierungsumgebung, Verhalten unter Last und Verhalten mit sehr grossen Datenmengen bewusst ausgeblendet.

4.4.2 Installierbarkeit

Ein Installationsskript ist nicht zu erstellen. Die Web-Anwendung kann für den Proof-of-Concept manuell auf dem Webserver deployed werden. Die BI-Komponenten werden ebenfalls manuell auf dem Datenbankserver der Lab-Umgebung installiert.

4.4.3 Anpassbarkeit

Es soll die Möglichkeit bestehen, mit möglichst geringem Aufwand neue, auf verschiedene Benutzerrollen angepasste Reports zu erstellen.

4.4.4 Stabilität

Die Software-Komponenten müssen einer Demonstration unter Alltagsbedingungen standhalten.

5 Konzepte DW

Dieses Kapitel beschreibt einige allgemeine Konzepte beim Aufbau von Data Warehouses (DW). DWs werden verwendet, um das Reporting und Analysieren von Daten zu vereinfachen. Hierbei versucht man häufig, die streng normalisierte, redundanzfreie, produktive Datenbank um ein DW zu erweitern, das weit performantere Zugriffe und eben auch Reporting über lange Zeiträume erlaubt. Daten werden kontrolliert redundant abgespeichert, um schnellere Zugriffe zu ermöglichen. Auf eine Normalisierung wird häufig verzichtet. Die Daten im DW werden aus einer oder mehreren produktiven Datenbanken oder auch Flat Files kopiert und mit zusätzlichen berechneten Daten angereichert, um die Berechnungszeiten bei der Abfrage der Daten klein zu halten. Im Gegenzug beanspruchen DWs dafür in der Regel sehr viel Speicherplatz.

5.1 Dimensionen

Der Prozess der Datenanalyse beschäftigt sich im Allgemeinen nicht nur mit den Detaildaten, wie sie aus den einzelnen Geschäftsvorfällen aufgezeichnet wurden, sondern mit konsolidierten, aggregierten Daten verschiedenster Ebenen. Konsolidierung von Daten ist der Prozess, in dem Teile von Informationen zu einzelnen Blöcken relevanten Wissens zusammengeführt werden. Die höchste Ebene im Pfad einer Datenkonsolidierung wird als Dimension dieser Daten bezeichnet. So können beispielsweise die Umsätze eines Unternehmens in verschiedener Hinsicht konsolidiert werden, beispielsweise nach dem Lieferziel, dem Produkt oder der Zeit[3]. In diesem Beispiel sind das Lieferziel, das Produkt und die Zeit jeweils eine Dimension.

5.2 Measures

Measures werden die eigentlichen Detaildaten aus den einzelnen Geschäftsvorfällen genannt. Im vorigen Beispiel wäre dies also der Umsatz.

5.3 Cubes

Dimensionen und Measures definieren die Struktur und den Inhalt einer multidimensionalen Datenmenge. Dies wird häufig als Cube bezeichnet. Für drei Dimensionen kann ein Cube visuell dargestellt werden. Folgende Abbildung zeigt einen Cube mit den Dimensionen *Route*, *Source* und *Time* und den Measures *Packages* und *Last*. Ein Cube kann aus beliebig vielen Dimensionen bestehen, wobei die unterschiedlichen Ausprägungen der Dimension den Ort im Cube beschreiben, wo die Werte (Measures) stehen.

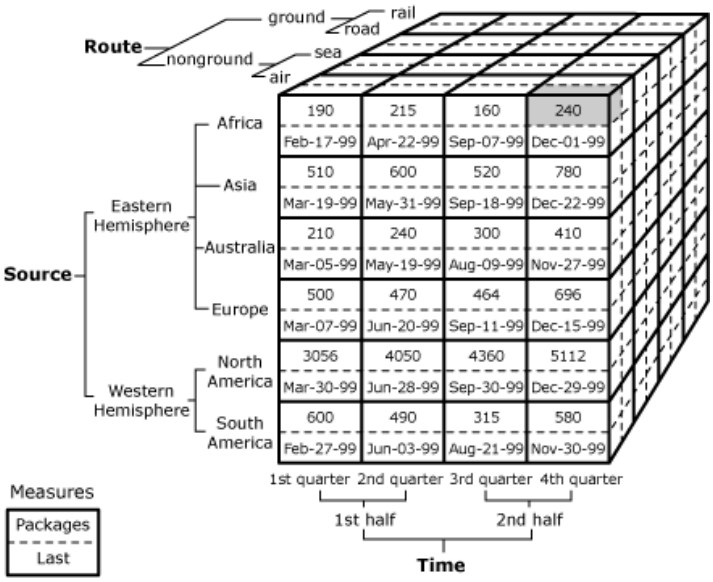


Abbildung 14: Beispiel eines Cubes[4]

[3] Quelle: (Brosius, Azevedo, Dehnert, Neumann, & Scheerer, BI & Reporting, S. 47, 2009)
[4] Quelle: ([BI] Resort, 2010)

5.4 Dimensions- und Faktentabellen

Um einen Cube zu definieren und mit Daten zu versorgen, müssen ihm die Dimensionen mit ihren Elementen und die Werte der Measures übergeben werden. Meistens wird dafür eine relationale Datenbank verwendet. Dabei gilt:

- Die Measures werden in einer sogenannten Faktentabelle abgespeichert.
- Die Elementwerte jeder Dimension werden in einer sogenannten Dimensionstabelle abgespeichert.
- Der Primärschlüssel jeder Dimensionstabelle erscheint als Fremdschlüssel in der Faktentabelle.

Diese Anordnung der Daten wird als Starschema bezeichnet und wird im nächsten Abschnitt genauer beschrieben.

Faktentabellen sowie Dimensionstabellen besitzen typische Eigenschaften. So ändern sich die Daten in der Faktentabelle typischerweise sehr häufig. Beispielsweise erhält eine Faktentabelle, welche die Verkäufe eines Unternehmens festhält, bei jedem Update hunderte oder tausende neue Einträge. So werden Faktentabellen meistens auch sehr gross. Sie besitzen eine grosse Anzahl an Datensätzen. Die Inhalte der Dimensionstabellen ändern sich weniger schnell. Die Verkäufe eines Unternehmens könnten zum Beispiel nach Filiale oder Datum konsolidiert werden. Die Eröffnung einer neuen Filiale ist aber doch ein eher seltenes Ereignis. Die Grösse der Dimension ist abhängig von dem, was sie darstellt. Beispielsweise können Zeitdimensionen auch sehr gross werden. Wenn beispielsweise über ein Jahr hinweg jeder Verkauf einer Zeit zugeordnet werden soll und das auf eine Minute genau, benötigt die Zeittabelle 60*24*365=525‘600 Datensätze.

Ein wichtiger Punkt und Stolperstein bei Dimensions- und Faktentabellen stellt auch die Tatsache dar, dass zu jedem Datensatz in der Faktentabelle jede der Dimensionen definiert sein muss. Es sind also keine NULL-Werte erlaubt, da sich der Primärschlüssel der Faktentabelle aus den Primärschlüsseln der Dimensionstabellen zusammensetzt.

5.5 Starschema

In diesem Beispiel eines Starschemas wird die Beziehung zwischen den Dimensions- und den Faktentabellen deutlich. Die Faktentabelle Fact_Sales enthält alle Fremdschlüssel der drei umliegenden Dimensionstabellen. In der Faktentabelle sind auch die Measures, in diesem Fall nur das Feld *Units_Sold*, untergebracht. Wie man sieht, sind in einem Starschema die Dimensionstabellen nicht normalisiert, das heisst, alle relevanten Daten zu einer Dimension werden in einer einzigen Tabelle abgespeichert.

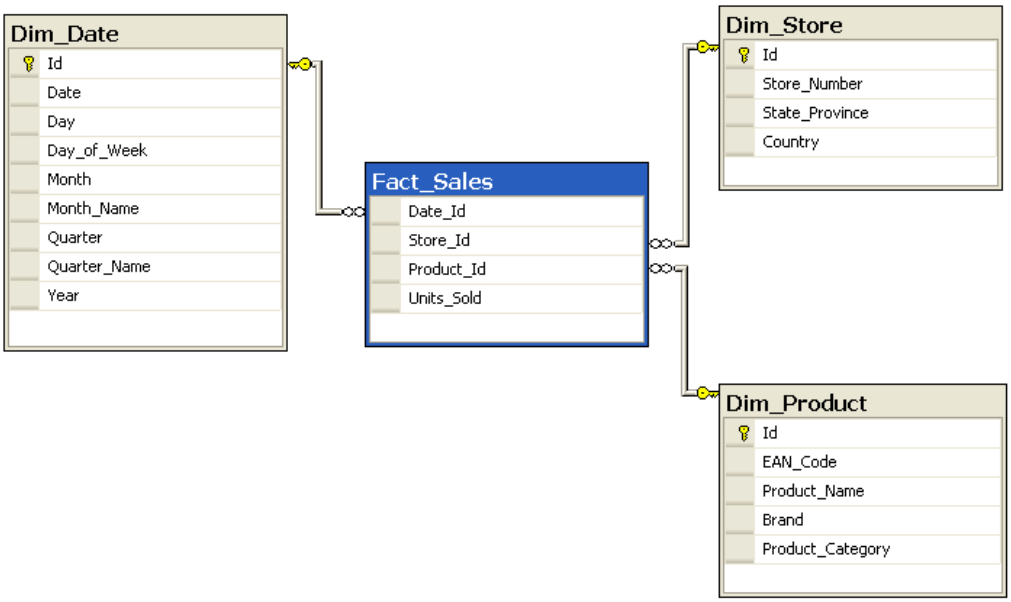


Abbildung 15: Beispiel eines Star Schemas[5]

[5] Quelle: (Wikipedia - Star Schema, 2010)

5.6 Snowflakeschema

Im Gegensatz zum Starschema sind beim Snowflakeschema pro Dimension mehrere Tabellen möglich. Der Grund ist der, dass die Daten in den Dimensionstabellen in der dritten Normalform abgelegt werden. Heutzutage wird das Starschema meist dem Snowflakeschema vorgezogen, da damit einfachere sowie performantere Abfragen möglich sind und keine Joins auf den einzelnen Dimensionen gemacht werden müssen.

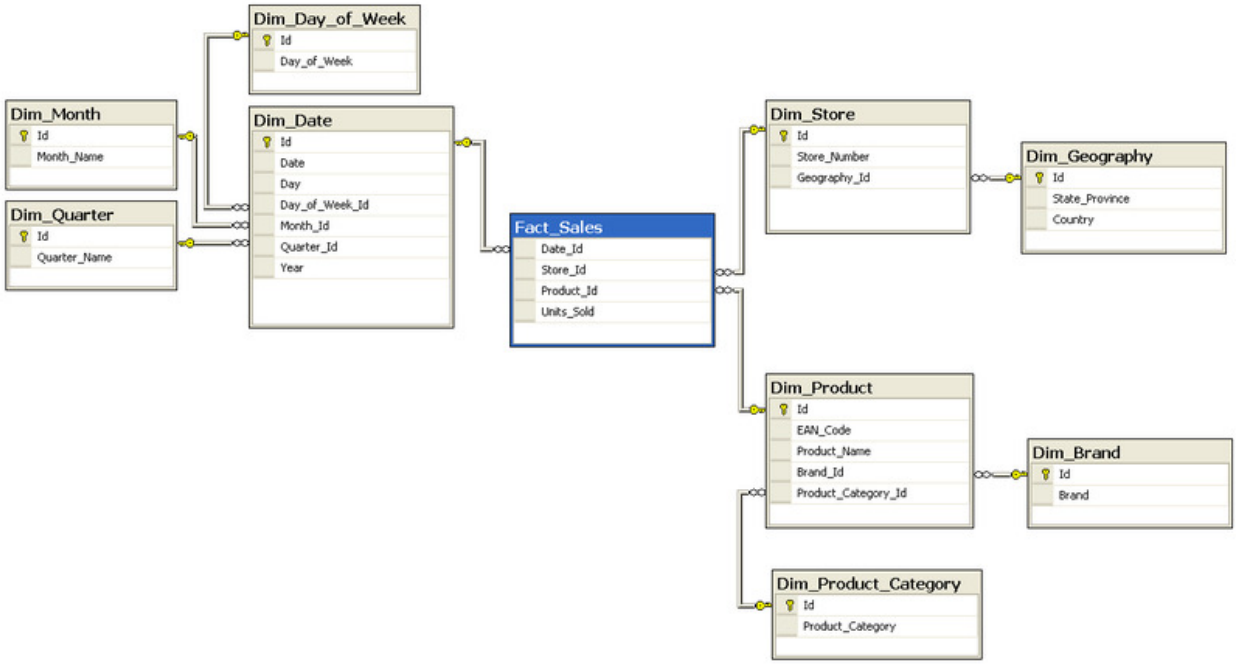


Abbildung 16: Beispiel eines Snowflake Schemas[6]

5.7 Galaxy Schema

In einem Galaxyschema sind mehrere Faktentabellen vorhanden, welche mit den gleichen Dimensionstabellen verknüpft sind. Man könnte auch sagen, dass ein Galaxyschema die Vereinigung mehrerer Starschemas ist, wobei die verschiedenen Starschemas die gleichen Dimensionstabellen benutzen. Dies ist ein sehr häufiger Fall, da Daten nach unterschiedlichen Aspekten ausgewertet werden können und somit auch unterschiedliche Starschemas bzw. Cubes erstellt werden, die auf den gleichen Dimensionen basieren.

5.8 MDX (Multidimensional Expressions)

MDX ist die Abfragesprache für OLAP Cubes und in vielerlei Hinsicht ähnlich zu SQL. Dieses Kapitel gibt eine kurze Einführung in die wichtigsten Möglichkeiten von MDX. MDX ist sehr komplex, darum kann hier bei Weitem nicht der ganze Sprachumfang erklärt werden. Wir beschränken uns auf die paar wichtigsten Konzepte, die auch für die Umsetzung des Projekts LUBI relevant waren.[7]

Zum Verständnis der Beispiele müssen erst einige Begriffe erklärt werden.

Ein Cube wird durch seine Attributhierarchien aufgespannt, die meistens aus den Spalten der Dimensionstabellen aus dem relationalen Modell (Starschema) gebildet werden. Eine Dimension ist sozusagen der Container für Attributhierarchien. Ein Tupel ist ein Ausdruck, der eine Zelle im Cube definiert. Tupel müssen nicht alle Achsen enthalten. Wenn Achsen weggelassen werden, verwendet Analysis Services implizit ein Element dieser Achsen. Dies kann ein Standardelement sein, das All-Element der Attributhierarchie, welches alle Elemente agreggiert oder einfach das erste Element der Attributhierarchie. Mengen sind Sammlungen von keinem, einem oder mehreren Tupeln.

[6] Quelle: (Wikipedia - Snowflake Schema, 2010)
[7] MDX Funktionsreferenz: http://msdn.microsoft.com/de-de/library/ms145595.aspx

5.8.1 Qualifizierte Namen

Qualifizierte Namen beinhalten den Namen des Objekts und aller übergeordneter Elemente.

Der qualifizierte Name einer Dimension ist nur ihr Name:
```
[Dim_Date]
```

Der qualifizierte Name einer Hierarchie besteht aus dem Namen der Dimension und der Hierarchie:
```
[Dim_Date].[DateHierarchy]
```

Der qualifizierte Name einer Ebene in der Hierarchie besteht aus der Dimension, allen übergeordneten Ebenen und der Ebene selbst:
```
[Dim Date].[DateHierarchy].[Year]
```

Das Element einer Hierarchie wird durch die Dimension, alle Ebenen der Hierarchie und das Element selbst bezeichnet:
```
[Dim Date].[DateHierarchy].[Year].&[2005].&[Q2].&[6].&[1]
```

Als nächstes folgen ein paar Beispiele zur Veranschaulichung von MDX. Sie stammen alle aus dem MSDN von Microsoft.[8]

5.8.2 SELECT

Eine MDX-Abfrage beginnt immer mit dem Schlüsselwort SELECT. Danach werden die Abfrageachsen spezifiziert. Nach dem Schlüsselwort FROM folgt dann der Name des Cubes, auf dem die Abfrage erfolgt. Optional kann dann noch eine WHERE-Klausel angegeben werden, deren Tupel oder Menge für Cubeachsen, die nicht in den Abfrageachsen enthalten sind, Elemente als Einschränkung vorgeben.

Folgendes Beispiel zeigt eine grundlegende MDX Abfrage:
```
SELECT
    {[Measures].[Sales Amount], [Measures].[Tax Amount] } ON COLUMNS,
    {[Date].[Fiscal].[Fiscal Year].&[2002],
     [Date].[Fiscal].[Fiscal Year].&[2003]} ON ROWS
FROM [Adventure Works]
WHERE ([Sales Territory].[Southwest])
```

Die Abfrage gibt ein Resultset zurück, das die Umsatz- (Sales Amount) und Steuerbeträge (Tax Amount) für 2002 und 2003 in den südwestlichen Vertriebsregionen enthält.

- Die SELECT-Klausel legt die Abfrage-Achsen auf die Elemente Sales Amount (Umsatz) und Tax Amount (Steuern) der Measures-Dimension sowie auf die Elemente 2002 und 2003 der Date-Dimension fest.
- Die FROM-Klausel zeigt an, dass die Datenquelle im Adventure Works-Cube besteht.
- Die WHERE-Klausel definiert das Southwest-Element der Sales Territory-Dimension zur Slicer-Achse.

Im oberen Beispiel wurden die Achsenaliase ON COLUMNS und ON ROWS verwendet. Die Achsen können jedoch auch durch Nummern identifiziert werden. Das folgende Beispiel ist also gleichbedeutend mit dem letzten.
```
SELECT
    {[Measures].[Sales Amount], [Measures].[Tax Amount] } ON 0,
    {[Date].[Fiscal].[Fiscal Year].&[2002],
     [Date].[Fiscal].[Fiscal Year].&[2003]} ON 1
FROM [Adventure Works]
WHERE ([Sales Territory].[Southwest])
```

5.8.3 Berechnete Elemente (Calculated Members)

Wenn in einer Abfrage Elemente benötigt werden, die im Cube nicht vorkommen, aber berechnet werden können, dann können berechnete Elemente verwendet werden. Dazu benutzt man das WITH-Keyword.

```
WITH
   MEMBER [Measures].[Special Discount] AS
      [Measures].[Discount Amount] * 1.5
SELECT
   [Measures].[Special Discount] ON COLUMNS,
   NON EMPTY [Product].[Product].MEMBERS ON ROWS
FROM [Adventure Works]
WHERE [Product].[Category].[Bikes]
```

[8] http://msdn.microsoft.com/de-de/library/

Hier wird ein berechnetes Element Special Discount aus dem Measure Discount Amount berechnet, welches sich bereits im Cube befindet. Dieses berechnete Element wird danach in der Abfrage verwendet.

5.8.4 Benannte Mengen (Named Sets)

In einer benannten Menge können interesannte Tupel gleicher Dimensionalität beschrieben und in der Abfrage verwendet werden. Benannte Mengen werden mittels WITH SET definiert. Das folgende Beispiel definiert ein benanntes Set von verschiedenen Weinen und benutzt dieses dann in der Abfrage.

```
WITH SET [ChardonnayChablis] AS
   {[Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and Wine].[Wine].[Good].[Good
      Chardonnay],
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and Wine].[Wine].[Pearl].[Pearl
      Chardonnay],
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and Wine].[Wine].[Portsmouth].[Portsmouth
      Chardonnay],
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and Wine].[Wine].[Top Measure].[Top
      Measure Chardonnay],
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and Wine].[Wine].[Walrus].[Walrus
      Chardonnay],
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and Wine].[Wine].[Good].[Good Chablis
      Wine],
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and Wine].[Wine].[Pearl].[Pearl Chablis
      Wine],
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and Wine].[Wine].[Portsmouth].[Portsmouth
      Chablis Wine],
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and Wine].[Wine].[Top Measure].[Top
      Measure Chablis Wine],
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and Wine].[Wine].[Walrus].[Walrus Chablis
      Wine]}

SELECT
   [ChardonnayChablis] ON COLUMNS,
   {Measures.[Unit Sales]} ON ROWS
FROM Sales
```

5.8.5 NONEMPTY

Die NONEMPTY Funktion gibt die Menge der nicht leeren Tupel einer angegebenen Menge zurück, basierend auf dem Kreuzprodukt der angegebenen Menge mit einer zweiten Menge.

Die folgende Abfrage enthält ein einfaches Beispiel für die Verwendung von NonEmpty. Es werden alle Kunden angezeigt, für die am 1. Juli 2001 Internet Sales Amount ungleich NULL ist.

```
SELECT [Measures].[Internet Sales Amount] ON 0,
NONEMPTY(
  [Customer].[Customer].[Customer].MEMBERS
, {([Date].[Calendar].[Date].&[20010701], [Measures].[Internet Sales Amount])}
)
ON 1
FROM [Adventure Works]
```

5.8.6 FILTER

Die FILTER Funktion filtert eine angegebene Menge basierend auf einer Suchbedingung und gibt dann das Resultset zurück.

Im folgenden Beispiel wird die Filter-Funktion auf der ROWS-Achse einer Abfrage verwendet, um nur Datumsangaben zurückzugeben, bei denen Internet Sales Amount größer als 10.000 US-Dollar ist:

```
SELECT [Measures].[Internet Sales Amount] ON 0,
FILTER(
  [Date].[Date].[Date].MEMBERS
, [Measures].[Internet Sales Amount]>10000)
ON 1
FROM
[Adventure Works]
```

5.8.7 COUNT

Die COUNT Funktion gibt die Anzahl der Zellen in einer Menge zurück.

Im folgenden Beispiel wird die Anzahl der Zellen in der Menge der Elemente bestimmt, die aus den untergeordneten Elementen der Model Name-Attributhierarchie in der Product-Dimension bestehen.

```
WITH MEMBER measures.X AS
   COUNT([Product].[Model Name].CHILDREN)
SELECT Measures.X ON 0
FROM [Adventure Works]
```

5.8.8 AVG

Die AVG Funktion wertet eine Menge aus und gibt den Durchschnitt der nicht leeren Werte der Zellen in der Menge zurück, gemittelt über die Measures in der Menge oder über ein angegebenes Measure.

Im folgenden Beispiel wird der Durchschnittswert für ein Measure über eine bestimmte Menge zurückgegeben. Beachten Sie, dass das angegebene Measure dem Standardmeasure für die Elemente der angegebenen Menge oder eines bestimmten Measures entsprechen kann.

```
WITH SET [NW Region] AS
  {[Geography].[State-Province].[Washington]
, [Geography].[State-Province].[Oregon]
, [Geography].[State-Province].[Idaho]}


MEMBER [Geography].[Geography].[NW Region Avg] AS
   AVG ([NW Region]
   --Uncomment the line below to get an average by Reseller Gross Profit Margin
   --otherwise the average will be by whatever the default measure is in the cube,
   --or whatever measure is specified in the query
   --, [Measures].[Reseller Gross Profit Margin]
)
SELECT [Date].[Calendar Year].[Calendar Year].Members ON 0
FROM [Adventure Works]
WHERE ([Geography].[Geography].[NW Region Avg])
```

5.8.9 MIN, MAX

Die Minimum- bzw. Maximumfunktionen funktionieren genauso wie die Durchschnittsfunktion AVG.

6 Technologieanalyse

6.1 Microsoft SQL Server 2008

Die folgende Abbildung zeigt eine Übersicht der mit Business Intelligence in Verbindung stehenden Services des SQL Server 2008. Die einzelnen Services werden im Folgenden überblickmässig beschrieben. Ausserdem folgt eine Beschreibung der eingesetzten Programme. Alle beschriebenen Programme bzw. Services sind im Lieferumfang von Microsoft SQL Server 2008 Enterprise Edition enthalten.

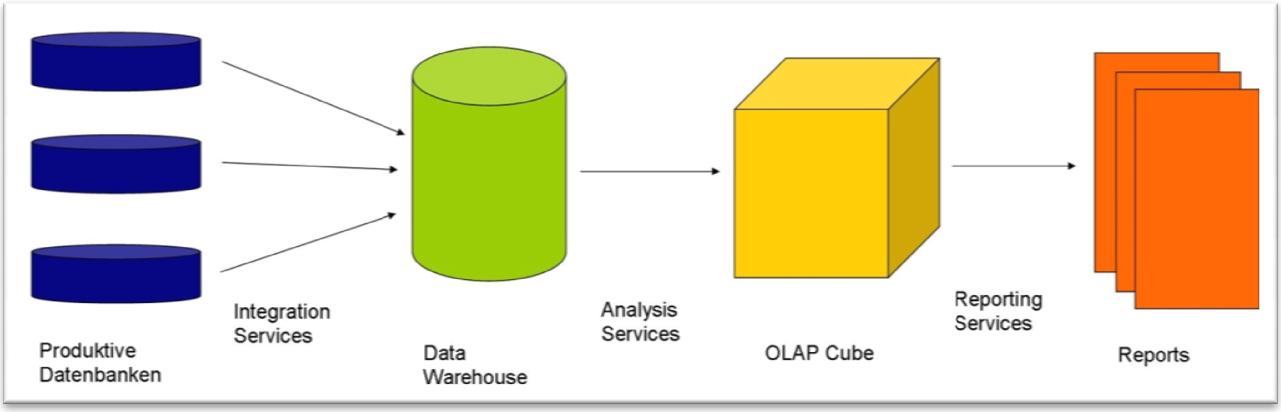


Abbildung 17: Übersicht SQL Server 2008 Business Intelligence Services

6.1.1 SQL Server Integration Services (SSIS)

Mittels der Integration Services lassen sich Datenintegrations- und Datentransformationslösungen realisieren. Als Quelle einer solchen Integration kann zum Beispiel eine produktive, relationale Datenbank dienen. Weitere Quellen wie zum Beispiel ein Flat File mit kommaseparierten Daten sind aber auch möglich. Die Daten werden aus der Datenquelle geholt (Extract), wenn nötig für das DW transformiert und optimiert (Transform) und danach im DW abgespeichert (Load). Dieses Prinzip wird auch als ETL bezeichnet. Der ganze Vorgang wird in einem sogenannten Package abgespeichert, welches dann z.B. über den SQL Server Agent regelmässig ausgeführt werden kann.

Das Ziel des DW ist es, die Daten für die Analysis Services möglichst optimiert zur Verfügung zu stellen.

6.1.1.1 Packages

Der ganze Vorgang der Überführung der Daten aus der Live Datenbank in das DW kann mittels Control Flows innerhalb eines Integration Service Projekts definiert werden. Ein Package besteht aus einem sogenannten Control Flow, welcher verschiedene Tasks enthalten kann. Ein Control Flow kann Data Flow Tasks enthalten, in welchen der eigentliche ETL-Vorgang stattfindet. Innerhalb eines Data Flow Tasks gibt es verschiedene Elemente, mit der die Daten extrahiert, transformiert oder gespeichert werden können (ETL).

Packages können abgespeichert und beispielsweise mit dem SQL Server Agent planmässig ausgeführt werden (z.B. einmal pro Tag).

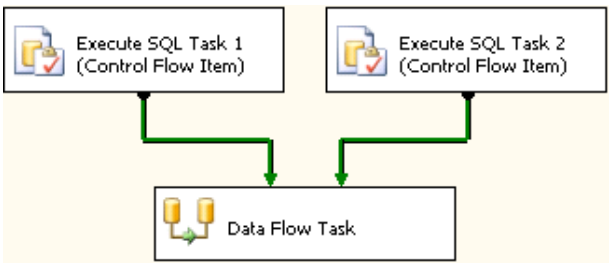


Abbildung 18: Beispiel Control Flow mit zwei SQL- und einem Data Flow Task

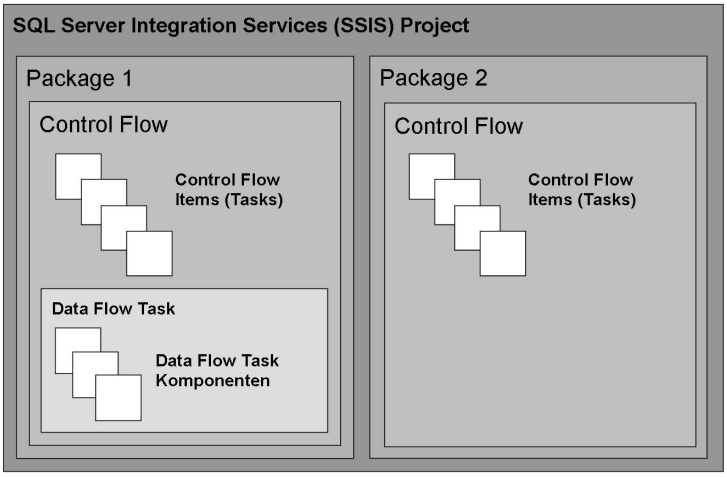


Abbildung 19: SSIS Übersicht (vereinfacht)

6.1.1.2 Control Flow Items (Tasks)

Die Control Flow Items steuern den Ablauf. Die verschiedenen Items werden grafisch miteinander verbunden, was die Reihenfolge bestimmt, in welcher die Tasks ausgeführt werden. Auch Verzweigungen und Schleifen sind selbstverständlich möglich. In diesem Abschnitt werden nur die für die BA verwendeten Items beschrieben, auch wenn Microsoft noch weitere zur Verfügung stellt.

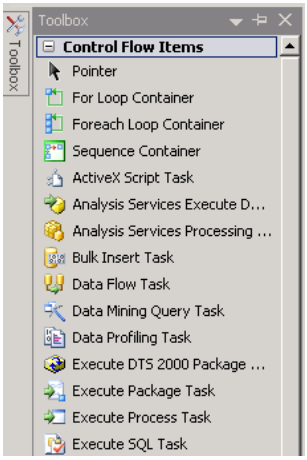


Abbildung 20: Control Flow Items (Ausschnitt)

6.1.1.2.1 Execute SQL Task

Diese Komponente führt einen oder mehrere SQL Statements auf einer Datenbank aus.

6.1.1.2.2 Analysis Services Processing Task

Mit dieser Komponente kann ein Processing des Cubes vorgenommen werden. Beim Processing eines Cubes holt dieser die aktuellen Daten von seiner Datenquelle. Die Komponente wird am sinnvollsten nach einem ETL-Vorgang verwendet, um die neuen Daten in den Cube zu bringen.

6.1.1.2.3 Data Flow Task

Dieser Task stellt einen sogenannten ETL-Vorgang dar. Ein Data Flow Task kann selbst wiederum mehrere Komponenten enthalten, die entweder von einer Datenquelle lesen, in eine Datensenke schreiben oder Daten transformieren.

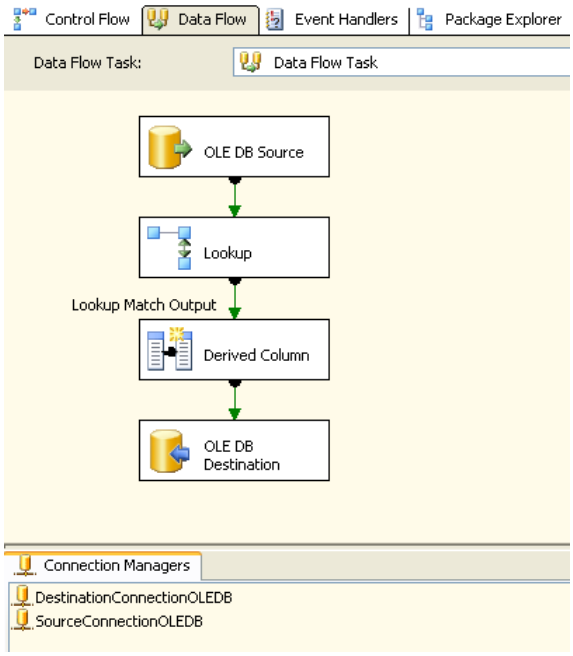


Abbildung 21: Beispiel Data Flow Task mit OLE DB Source/Destination sowie zwei Transformations-Komponenten

Ein Data Flow Task ist ein Task innerhalb des Control Flows und kann als Unterpaket angesehen werden. Hier findet der eigentliche ETL-Vorgang statt. Es wird eine Source definiert, von der die Daten gelesen werden, die Daten werden anschliessend den Bedürfnissen entsprechend transformiert und zum Schluss in eine Destination geschrieben. Die verschiedenen Komponenten eines Data Flow Tasks werden grafisch miteinander verbunden. Diese Verbindungen stellen den Datenfluss dar, der mit jeder Komponente erweitert werden kann.

Um das Konzept des Datenflusses zu veranschaulichen, hilft folgende Darstellung.

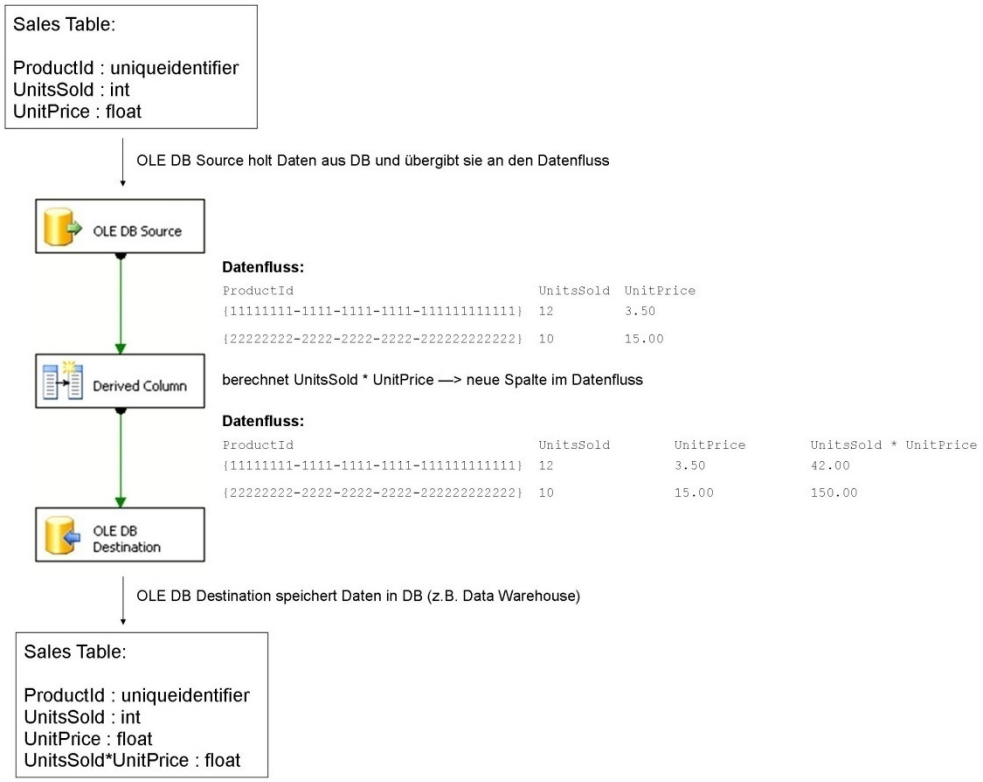


Abbildung 22: Veranschaulichung Datenfluss

Jeder Data Flow Task besteht aus mindestens einer Data Flow Source und einer Data Flow Destination, ansonsten wäre er sinnlos. Die Anzahl dieser Komponenten kann allerdings beliebig vergrössert werden. Zwischen einer Source und einer

Destination können beliebig viele Transformationen stattfinden und es sind nebenbei auch beliebig viele Sources und Destinations erlaubt.

In der obigen Veranschaulichung greift die OLE DB Source auf die Tabelle *Sales* einer Datenbank zu, liest alle Datensätze aus und fügt sie dem Datenfluss hinzu. Die Transformationskomponente *Derived Column* erweitert den Datenfluss um die Spalte *UnitsSold * UnitPrice*, welche aus bereits vorhandenen Spalten im Datenfluss berechnet wird. Die OLE DB Destination schreibt die Daten aus dem Datenfluss abschliessend in eine Tabelle der Zieldatenbank. Das Ziel könnte beispielsweise ein DW sein, das für Datenanalysen verwendet wird.

6.1.1.3 Data Flow Task Komponenten

Dieser Abschnitt beschreibt die für die BA verwendeten Komponenten eines Data Flow Tasks. Microsoft stellt noch weitere Komponenten sowie die Möglichkeit zur Verfügung, mittels einer Skriptkomponente selber Komponenten zu entwerfen. Als nächstes werden die beiden Komponenten OLE DB Source sowie OLE DB Destination beschrieben, die den Extract- bzw. Load-Teil des ETL-Vorgangs darstellen. Danach folgen Beschreibungen mehrerer Komponenten, die für die Datentransformation eingesetzt werden, also dem Transform-Teil des ETL-Vorgangs entsprechen.

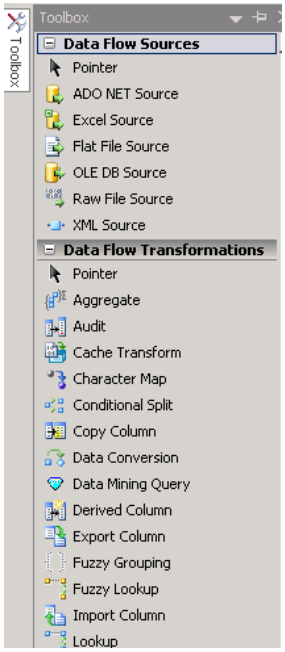


Abbildung 23: Data Flow Task Komponenten (Ausschnitt)

6.1.1.3.1 Komponente OLE DB Source

Die Komponente OLE DB Source erlaubt es, Daten aus einer relationalen Datenbank zu lesen und dem Datenfluss hinzuzufügen. Beim Konfigurieren der OLE DB Source ist es möglich, eine Tabelle einer Datenbank sowie die gewünschten Spalten auszuwählen. Weiter muss ein sogenannter Connection Manager angegeben werden, der vorher angelegt werden muss und den Zugriff auf die Datenbank spezifiert. Die OLE DB Source liest alle Datensätze der gewählten Tabelle und übergibt diese dem Data Flow, wobei nur die ausgewählten Spalten mitgegeben werden.

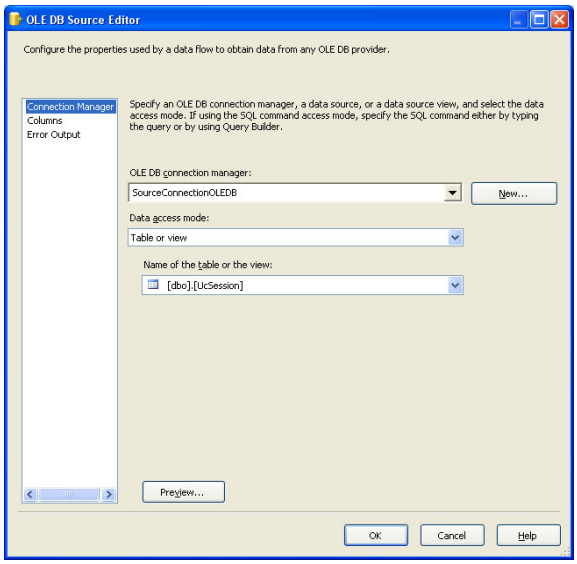


Abbildung 24: OLE DB Source Editor, Seite *Connection Manager*

6.1.1.3.2 Komponente OLE DB Destination

Die Komponente OLE DB Destination erlaubt es, Daten aus dem Datenfluss zu entnehmen und in eine relationale Datenbank zu speichern. Wie bei der OLE DB Source muss auch hier ein Connection Manager angegeben werden, der angibt, in welche Datenbank die Daten aus dem Datenfluss gespeichert werden. Ausserdem muss auf der Seite *Mappings* angegeben werden, welche Spalten aus dem Datenfluss auf welche Spalten in der Datenbanktabelle abgebildet werden.

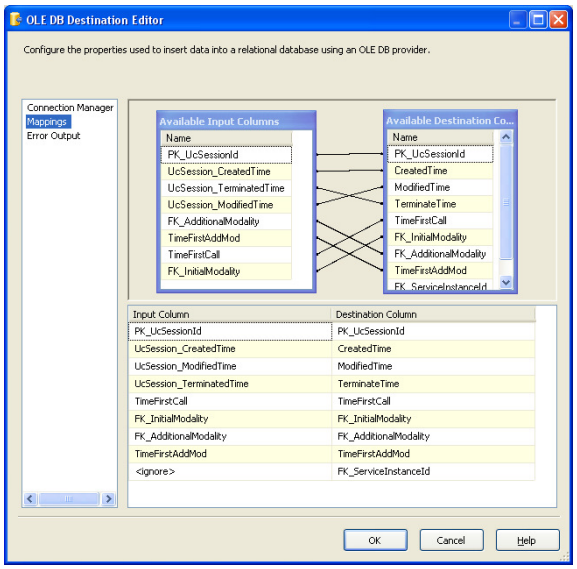


Abbildung 25: OLE DB Destination, Seite *Mappings*

6.1.1.3.3 Komponente Lookup

Mit der Komponente Lookup werden die Zeilen des Datenflusses mit den Tupeln einer Tabelle einer Datenbank verglichen und aufgrund von Übereinstimmungen einer Spalte oder mehrerer Spalten zusammengefügt. Dieser Vorgang ist vergleichbar mit einem Join, wobei der Datenfluss die linke Tabelle und die Datenbanktabelle die rechte Tabelle des Joins darstellt. Es kann ausgewählt werden, welche Spalten der rechten Tabelle dem Datenfluss hinzugefügt werden. Die Komponente Lookup hat einen Match Output sowie einen No Match Output, so dass die Zeilen mit einem Treffer (Match) und diejenigen ohne Treffer getrennt werden können. Alternativ können alle Zeilen dem Match Output zugeordnet werden. In diesem Fall werden bei Zeilen, in denen keine Übereinstimmung mit der rechten Tabelle gefunden wird, Nullwerte für die Spalten der rechten Tabelle eingefügt. Eine dritte Konfigurationsmöglichkeit ist das Fehlschlagen der gesamten Komponente, wenn mindestens eine Zeile keine Übereinstimmung erzielt.

6.1.1.3.4 Komponente Derived Column

Die Komponente Derived Column erweitert den aktuellen Datenfluss um weitere Spalten, indem neue Werte aus bereits vorhandenen Spalten berechnet werden. Dabei können Variablen, Ausdrücke und Funktionen eingesetzt werden.

6.1.1.3.5 Komponente Conditional Split

Die Komponente Conditional Split hat mehrere Ausgänge. Es können Bedingungen festgelegt werden, unter denen Zeilen des Datenflusses einem bestimmten Ausgang zugeordnet werden. So können bestimmte Zeilen des Datenflusses gezielt aussortiert und anschliessend auch unterschiedlich behandelt werden.

6.1.1.3.6 Komponente Sort

Die Komponente Sort schafft die Möglichkeit, die Zeilen im Datenfluss nach einer oder mehreren Spalten zu sortieren.

6.1.1.3.7 Komponente Merge Join

Mit dieser Komponente können mehrere Datenflüsse zusammengeführt werden. Der Jointyp kann entweder auf Inner Join, Left Outer Join oder Full Outer Join gestellt werden. Konfigurierbar sind die Spalten, auf denen der Join auf Gleichheit prüft. Weiter kann eingestellt werden, welche Spalten beider Datenflüsse an den Ausgang der Komponente weitergeleitet werden sollen.

6.1.1.4 Connection Manager

Connection Manager werden definiert, um Servername, Datenbank und Authentifizierung für den Zugriff auf eine Datenquelle oder -senke zu speichern. Der Vorteil von Connection Managern ist, dass bei mehrmaligem Zugriff auf die gleiche Datenbank an mehreren Stellen im Datenfluss bzw. in mehreren Datenflüssen jeweils nur der Connection Manager angegeben werden muss und die nötigen Informationen für den Zugriff somit zentral gespeichert werden können.

6.1.1.5 Variablen

In Integration Services-Paketen können Werte und Objekte in Variablen zwischengespeichert werden. In vielen Tasks und Komponenten kann auf Variablen zugegriffen werden. Die meisten Tasks und Komponenten erlauben allerdings nur einen lesenden Zugriff.

6.1.2 SQL Server Analysis Services (SSAS)

Mittels der Analysis Services wird ein OLAP Cube (Online Analytical Processing) erstellt, welcher eine schnelle Analyse der Daten erlaubt. Die Daten beziehen die Analysis Services z.B. aus einer relationalen Datenbank.

6.1.2.1 Datenquellen (Data Sources)

Innerhalb eines Analysis Services Projekts muss eine Datenquelle definiert sein, aus der die multidimensional aufzubereitenden Daten stammen. Beim Erstellen einer Datenquelle wird ein Connection Manager erstellt, der die Metadaten hält, welche für den Zugriff auf die Datenquelle benötigt werden. Konkret kann ein sogenannter Provider, der Servername, die Authentifizierungsart und die Datenbank angegeben werden.

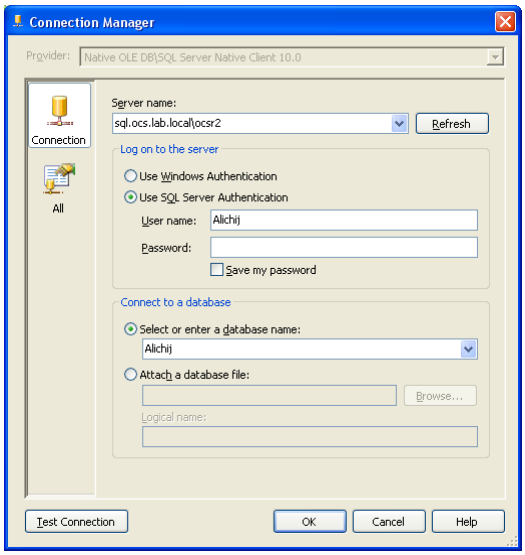


Abbildung 26: Connection Manager Einstellungen

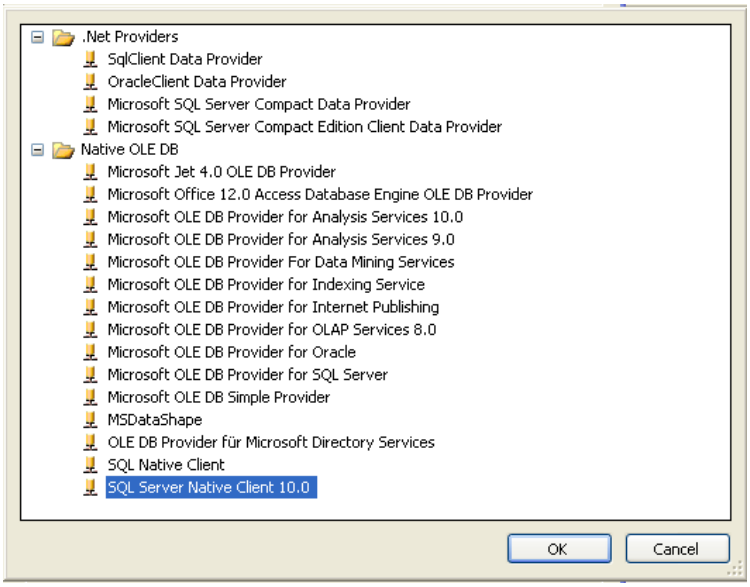


Abbildung 27: Provider

6.1.2.2 Datenquellensichten (Data Source Views)

Zusätzlich zu einer Datenquelle benötigt ein Analysis Services Projekt auch noch eine Datenquellensicht. DieDatenquellensicht bietet unter Anderem den Vorteil, dass Daten aus mehr als einer Datenquelle zusammengeführt (integriert) werden können. Dadurch können Cubes erstellt werden, die Daten aus mehreren Quellen integrieren.

6.1.2.3 Cubes

In den Analysis Services können Cubes anhand der Tabellen in der Datenquellensicht erstellt werden. Es ist zwar auch möglich, den Cube ohne Datenquelle und Datenquellensicht zu erstellen, so dass der Cube zuerst nur Metadaten enthält. Die Datenquelle und Datensichten müssen aber spätestens erstellt werden, wenn Daten im Cube angezeigt werden sollen.

6.1.2.4 Cubes bearbeiten

Die Cubes des Projekts befinden sich standardmässig immer im Ordner *Cubes* im Solution Explorer. Wenn der Cube im Solution Explorer des Visual Studio ausgewählt wird, erscheinen am oberen Bildschirmrand mehrere Registerkarten. Folgende Tabelle erklärt alle für das Projekt LUBI relevanten Registerkarten:

Registerkarte	Beschreibung
Cubestruktur (Cube Structure)	Die Registerkarte beinhaltet mehrere Bereiche. Der grösste Bereich ist die Datenquellensicht (Data Source View) mit einer Übersicht aller Fakten- bzw. Dimensionstabellen inkl. aller Fremdschlüsselbeziehungen (Implementierung mit Screenshot im Kapitel 10.3.3.1). Weiter gibt es einen Bereich *Measures*, in dem Measures erstellt und bearbeitet werden können. Beispielsweise ist für jedes Measure einstellbar, wie es standardmässig aggregiert wird. Dann gibt es noch einen Bereich Dimensionen, der es erlaubt, dem Cube Dimensionen hinzuzufügen, bzw. diese vom Cube zu entfernen.
Dimensionsverwendung (Dimension Usage)	Über diese Registerkarte lassen sich Beziehungen zwischen Faktentabellen (Measure Groups) und Dimensionen konfigurieren. Für jede Faktentabelle kann ein Attribut angegeben werden, welches als Fremdschlüssel in eine Dimensionstabelle fungiert. (Implementierung mit Screenshot in Kapitel 10.3.3.2)
Browser	Der Cubebrowser erlaubt das Durchsuchen des Cubes. Vorher müssen allerdings bestimmte Dimensionen und Measures in dafür vorbestimmte Bereiche gezogen werden. Da es sich hier nur um einen grafischen Browser handelt ohne die Möglichkeit, irgendwelche MDX Abfragen abzusetzen, ist dieser sehr limitiert. Für komplexere Abfragen kann das SQL Server Management Studio verwendet werden.

Tabelle 14: Cube Registerkarten

6.1.2.5 Dimensionen bearbeiten

Die Dimensionen des Projekts befinden sich immer im Ordner *Dimensionen* im Solution Explorer. Beim Auswählen einer Dimension werden ähnlich wie beim Cube verschiedene Registerkarten angezeigt. Folgende Tabelle erklärt wiederum die für das Projekt Wichtigen:

Registerkarte	Beschreibung
Dimensionsstruktur (Dimension Structure)	Die Registerkarte beinhaltet mehrere Bereiche. Der Bereich *Datenquellensicht* beinhaltet die der Dimension zugrunde liegenden Tabellen mit allen Attributen. Ein eigener Bereich *Attribute* dient zum Bearbeiten der Attribute. Beispielsweise können hier Attribute für eine Parent-Child-Hierarchie definiert wird. Vor dem Verwenden der Attribute im Bereich *Attribute* müssen diese allerdings zuerst noch vom Bereich Datenquellensicht dahin gezogen werden. Der Bereich *Hierarchien* schliesslich dient zum Erstellen von Hierarchien.
Attributbeziehungen (Attribute Relationships)	Diese Registerkarte zeigt das Attributbeziehungsdiagramm an, was vor allem bei Dimensionen mit Hierarchien interessant ist.
Browser	Der Dimensionsbrowser erlaubt das Durchsuchen der Dimension. Dies ist vor Allem bei Dimensionen mit Hierarchien interessant, da in diesem Fall Drilldowns möglich sind.

Tabelle 15: Dimension Registerkarten

6.1.2.6 Hierarchien

Auf den Dimensionen eines Cubes können Hierarchien erzeugt werden, um sogenannte Drilldowns zu ermöglichen. Drilldown nennt sich ein Verfahren, bei dem Informationen auf unterschiedlichen Detailstufen betrachtet werden. Je nach Detaillierungsstufe werden Daten ein- oder ausgeblendet. Wenn Zeilen ausgeblendet werden, werden auf der übergeordneten Stufe häufig aggregierte (zusammengefasste) Daten angezeigt. Ein häufiges Beispiel ist eine Zeithierarchie, bei der die verschiedenen Stufen Jahr (Year), Quartal (Quarter of Year), Monat (Month Of Year) und Tag (Day Of Month) sein könnten. Zu Beginn sieht man nur die Jahre und die detaillierteren Daten sind ausgeblendet. Durch Drücken des Pluszeichens

kann ein bestimmtes Jahr genauer untersucht werden, so dass dann die Quartale dieses Jahres angezeigt werden. Dieser Vorgang (Drilldown) kann fortgesetzt werden, bis man einen einzelnen Tag sieht.

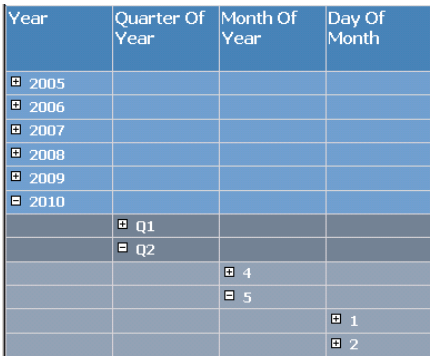


Abbildung 28: Beispiel eines Drilldowns

6.1.2.6.1 Parent-Child-Hierarchien

Ein Spezialfall einer Hierarchie ist die sogenannte Parent-Child-Hierarchie. Ein Merkmal der Parent-Child-Hierarchie ist, dass nicht jedes Blattelement (Elemente der untersten Hierarchieebene) die gleiche Anzahl von Ebenen über sich haben. Ein Beispiel wäre eine Personalhierarchie. Möglicherweise hat ein Vorgesetzter eines Mitarbeiters einen weiteren Vorgesetzten, ein anderer Mitarbeiter ist aber direkt dem obersten Vorgesetzten unterstellt. Diese Hierarchien werden auch als unausgeglichen bezeichnet, da die Anzahl Ebenen nicht vorgegeben ist. Bei Parent-Child-Hierarchien funktioniert das Schlüsselattribut der Dimension als Child, ein weiteres Attribut muss als Parent definiert werden.

6.1.2.7 Cube Processing

Wenn Änderungen an der dem Cube zugrunde liegenden Datenbank gemacht werden, also beispielsweise neue Daten eingefügt oder gelöscht wurden oder die Cubestruktur geändert wurde, muss ein Cube Processing durchgeführt werden. Beim Cube Processing werden die Dimensions- und Faktentabellen gelesen, aggregierte Daten berechnet und die Resultate im Cube gespeichert. Nach dem Cube Processing können Benutzer den Cube abfragen. Ein Cube Processing kann ausgelöst werden, in dem im Solution Explorer der Cube rechtsgeklickt und danach *Process...* gewählt wird.

6.1.3 SQL Server Reporting Services (SSRS)

Die Reporting Services erlauben auf der Basis des OLAP Cubes das dynamische Generieren von Reports. Die Reporting Services bieten die Möglichkeit, die Reports in beliebiger Form (siehe unten) und zeitlich gesteuert per E-Mail zu versenden, in eine Datenfreigabe zu speichern oder in eine Sharepoint-Bibliothek zu integrieren. Dazu sind teilweise spezielle Konfigurationen der Reporting Services nötig.

6.1.3.1 Export-Möglichkeiten Reporting-Services

Reports welche für die Microsoft SQL Reporting Services erstellt werden, liegen im RDL-Format vor. RDL ist die Abkürzung für Reporting Definition Language[9] und ist die XML-Repräsentation einer Report-Definition für die Reporting Services. RLD-Dateien bestehen typischerweise aus drei Sektionen: Einem Teil für Parameter und Datenbankverbindungen, einem Teil für die Felddefinitionen und einem Teil für die Seitengestaltung. Die RDL-Definition wurde mit der Intention, eine gemeinsame, interoperable Sprache für die kommerziellen Reportingsysteme zu schaffen, entwickelt. Da RDL wie bereits erwähnt auf XML basiert, kann sie problemlos durch den Entwickler erweitert oder angepasst werden. Der Namespace-URI findet sich unter http://schemas.microsoft.com/sqlserver/reporting/yyyy/mm/reportdefinition.

Diese so geschaffenen Reporte können zusammen mit den im OLAP-Cube vorliegenden Daten in diverse verschiedene Formate exportiert werden. Im Folgenden werden diese Formate kurz vorgestellt.

[9] Weitere Informationen: http://msdn.microsoft.com/en-us/library/ms155062.aspx

6.1.3.1.1 CSV (comma seperated value)

Der CSV-Export rendert den Report in eine flache Repräsentation der Daten des Original-Reports. Die Werte werden mit Trennzeichen, standartmässig Kommas, getrennt und sind dadurch einfach lesbar und von vielen verschiedenen Applikationen zu lesen. Es liegt in der Natur von CSV-Dateien, dass natürlich keine Charts, Bilder oder ähnliches exportiert werden kann. Auch hierarchische und gruppierte Daten können im CSV-Format nicht abgebildet werden und müssen deshalb vor dem Export plattgemacht werden.[10]

6.1.3.1.2 Microsoft Excel

Reporte können auch in das proprietäre Format von Microsoft Excel exportiert werden. Das erstellte File ist mit Microsoft Excel 97 und neuer kompatibel. Das Layout des Reports wird soweit wie möglich in Excel übernommen. Auch die meisten interaktiven Elemente werden von Excel beherrscht. Zu beachten gibt es die Limitationen von Excel, welche vor allem aus der beschränkten Anzahl von Spalten und Zeilen bestehen.[11]

6.1.3.1.3 Microsoft Word

Es besteht weiter die Möglichkeit, den Report in das ebenfalls proprietäre Format von Microsoft Word zu exportieren. Die Kompatibilität des erstellten Files erstreckt sich auf die Versionen Word 2000 und neuer. Der Report wird in Word in eine Tabellenstruktur gepackt. Bilder und Charts werden als Bilder eingefügt. Word unterstützt wenige interaktive Elemente welche aus dem Original-Report übernommen werden.[12]

6.1.3.1.4 HTML/MHTML

Der HTML-Export ist der Standart-Export, welcher auch auf dem Reporting-Server verwendet wird. Er bietet die gesamte Palette an interaktiven Funktionen (Drill-Downs, Suche, etc.). Desweiteren besteht die Möglichkeit, einen Snapshot des Reports (die gerade angezeigten Ebenen, etc.) als MHTML-Datei zu exportieren. MHTML steht für MIME HTML und ist eine Webpage-Archiv-Format (http://en.wikipedia.org/wiki/MHTML). Dabei werden die Ressourcen wie Bilder, Charts oder ähnliches zusammen mit dem HTML-Code in eine Multipart-MIME-Datei gepackt. Damit könnte z.B. der Report per E-Mail versandt werden ohne dass der Empfänger Zugriff auf Ressourcen auf dem Reporting-Server haben muss.[13]

6.1.3.1.5 XML

Mittels des XML-Exports können die Report-Daten in einer XML-Struktur für die Weiterverwendung in anderen Applikationen oder in Datenbanken aus den Reporting Services exportiert werden. Das verwendete XML-Schema hängt von dem zu exportierenden Report ab. Layout-Informationen werden keine exportiert, auch Bilder, Charts oder ähnliches werden ignoriert. Falls gewünscht können serverseitig XSL-Transformationen vorgenommen werden.[14]

6.1.3.1.6 Image

Der Report kann auch als Bild in verschiedensten Formaten exportiert werden. Hierbei geht jegliche Interaktivität verloren. Mehrseitige Reporte werden ausser im TIFF-Format in mehrere Bilder abgelegt.[15]

6.1.3.1.7 PDF

Die Reporting Services ermöglichen auch einen Export des Reports als PDF. Dies ist insbesondere für das Drucken und weiterverbreiten bestimmter Reports nützlich. Einige interaktive Elemente (Links, Document Maps, etc.) werden auch von PDF's unterstützt und stehen dem Benutzer zur Verfügung. [16]

[10] Weitere Informationen: http://msdn.microsoft.com/en-us/library/ms155919.aspx
[11] Weitere Informationen: http://msdn.microsoft.com/en-us/library/ms159836.aspx
[12] Weitere Informationen: http://msdn.microsoft.com/en-us/library/cc627455.aspx
[13] Weitere Informationen: http://msdn.microsoft.com/en-us/library/ms156022.aspx
[14] Weitere Informationen: http://msdn.microsoft.com/en-us/library/ms156430.aspx
[15] Weitere Informationen: http://msdn.microsoft.com/en-us/library/ms159216.aspx
[16] Weitere Informationen: http://msdn.microsoft.com/en-us/library/ms159713.aspx

6.1.3.2 Einbinden von Reports in eigene Applikationen

6.1.3.2.1 Einbindung als Steuerelement

Über die Visual Studio-Entwicklungsumgebung kann ein ReportViewer-Steuerelement wahlweise in eine WinForms- oder WebForms-Applikation integriert werden. Beide Elemente beherrschen sowohl den lokalen wie auch den Remote-Modus. Beim lokalen Modus werden berichte auf dem lokalen PC verarbeitet auf welchem nicht zwingend die Reporting Services installiert sein müssen. Im Remote-Modus wird das Rendern des Reports vom Reporting Server übernommen und nur das Ergebnis an den Client geliefert. Dieser Modus ist in der Regel der bevorzugte Modus weil alle Features des Reporting-Servers genutzt werden können.

Für die neuen Technologien WPF (Windows Presentation Foundation) und Silverlight existieren bis jetzt keine Steuerelemente für die SSRS. Jedoch können in die neuen Technologien Steuerelemente der älteren Technologien integriert werden. Dabei entstehen aber gewisse Unschönheiten, die es zu beachten gibt. So öffnet sich z.B. in Silverlight ein neues Fenster für jeden Report und in WPF integrierte WinForms können z.B. Probleme machen beim Traversieren der Applikation mittels der Tab-Taste.[17]

6.1.3.2.2 Einbindung mittels SOAP

Der Reporting-Server stellt mit der SOAP-API mehrere Webdienst-Endpunkte für die Interaktion mit den Reporting Services zur Verfügung. Es gibt Endpunkte für die Verwaltung des Servers (Eine für regulär konfigurierte und eine für Sharepoint konfigurierte Reporting Services) sowie einen Endpunkt für die Ausführung. Da über die SOAP-Schnittstellen sämtliche Verwaltungsfunktionen verfügbar sind, wird die SOAP-API insbesondere für die Verwaltung der Reporting Services aus einer eigenen Applikation heraus eingesetzt. Für die Anzeige der Reporte wird in der Regel auf den direkten URL-Zugriff gesetzt, da dieser performanter und einfacher zu realisieren ist. Dies weil unter anderem das Marshalling in und aus einem SOAP-Request entfällt. Auch steht über die SOAP-Schnittstelle keine Reportviewer-Toolbar zur Verfügung und der Entwickler muss selber eine Eingabe-Möglichkeit für Parameter, Exporteformate u.ä. bereitstellen.[18]

6.1.3.2.3 Integration über URL-Zugriff

Mittels Aufruf der Report-URL kann der Report in eine eigene Applikation eingebunden werden. Über Adress-Parameter können Report-Parameter und Angaben zur Customization an den Report Server übermittelt werden. Die Reports werden dann in gewünschter Form an die eigene Applikation geschickt. Dies funktioniert sowohl für Web-Applikationen wie auch für Desktop-Applikationen.[19]

6.1.3.3 Reporting Services Configuration Manager

Der Reporting Services Configuration Manager dient, wie der Name schon sagt, zum Konfigurieren der Reporting Services. Das Programm lässt sich standardmässig über Start/Programme/Microsoft SQL Server 2008/Configuration Tools/Reporting Services Configuration Manager starten.

17 Weitere Informationen: http://msdn.microsoft.com/de-de/library/aa337090.aspx
18 Weitere Informationen: http://msdn.microsoft.com/de-de/library/ms153968.aspx
und http://msdn.microsoft.com/de-de/library/ms155089.aspx
19 Weitere Informationen: http://msdn.microsoft.com/en-us/library/ms155362.aspx

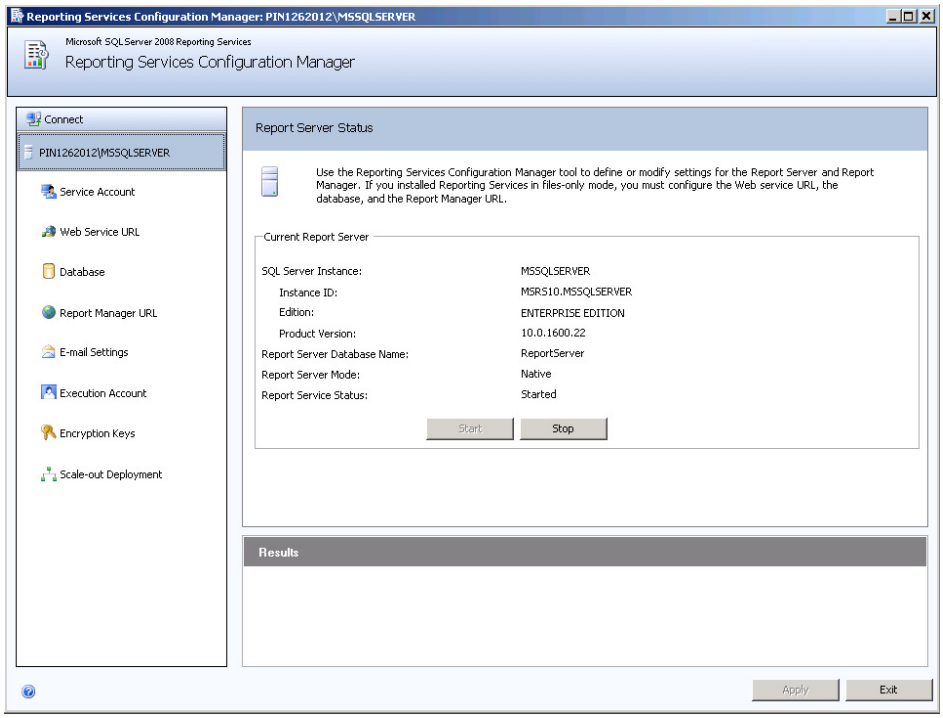


Abbildung 29: Reporting Services Configuration Manager

Folgende Einstellungen können vorgenommen werden:

- **Berichtsserverstatus (Report Server Status):** Hier kann der Reporting Services-Dienst gestartet werden. Ausserdem ist ersichtlich, ob der Dienst gerade läuft.
- **Dienstkonto (Service Account):** Hier kann angegeben werden, unter welchem Konto der Berichtsserver-Dienst laufen soll.
- **Webdienst-URL (Web Service URL):** Die Webdienst-URL kann unter diesem Punkt eingestellt werden. Sie bestimmt, unter welchem Pfad der Berichtsserver und der Berichtsmanager über das http-Protokoll erreicht werden kann.
- **Datenbank (Database):** Hier kann eingestellt werden, wie der Berichtsserver auf die Berichtsserver-Datenbank zugreift. Die Datenbank kann hier auch erstellt werden, falls dies noch nicht geschehen ist.
- **Berichts-Manager-URL (Report Manager URL):** Hier wird die Adresse des Berichtsmanagers eingestellt.
- **E-Mail-Einstellungen (E-Mail Settings)**: Um Reports per E-Mail zu verschicken, muss hier eine Absenderadresse sowie ein SMTP-Server definiert werden.
- **Ausführungskonto (Execution Account):** Hier kann ein Konto angegeben werden, unter dem der Berichtsserver spezielle Operationen, wie zum Beispiel die Ablage von Berichten im Dateisystem, ausführt.
- **Verschlüsselungsschlüssel (Encryption Keys):** Hier können die Schlüssel für die Verschlüsselung verwaltet und verschlüsselte Inhalte gelöscht werden.

6.1.3.4 Freigegebene Datenquelle

Wie die Integration Services- und Analysis Services Projekte brauchen auch die Reporting Services Projekte eine Datenquelle. Beim Definieren der Datenquelle werden ein Datenquellentyp, eine Verbindungszeichenfolge sowie Anmeldeinformationen angegeben.

6.1.3.5 Reports

Für das Erstellen der Reports steht ein Assistent zur Verfügung, der mit einem Rechtsklick auf den Ordner *Reports* des SSRS Projekts im Solution Explorer gestartet werden kann. Beim Durcharbeiten des Assistents muss eine Datenquelle angegeben werden, von der die Reportdaten geholt werden können. Danach ist es möglich, mit einem Abfrageeditor Abfragen auf der Datenquelle auszuführen. Diese Abfrageeditoren gibt es für relationale Datenbanken sowie OLAP Cubes. Einfachere Abfragen können grafisch zusammengeklickt werden, kompliziertere jedoch verlangen ein manuelles Bearbeiten des Abfrage-Statements (z.B. MDX oder SQL). Weiter kann innerhalb des Assistents noch die Darstellung des Reports eingestellt werden.

Drilldowns können aktiviert oder deaktiviert werden. Nach dem Beenden des Assistenten werden die Reports standardmässig im Ordner *Reports* gespeichert.

6.1.3.6 Berichtsdatasets

Wenn ein Report mittels des Assistenten erstellt wird, wird automatisch ein neues Berichtsdataset zur gewählten Datenquelle hinzugefügt. Pro Datenquelle sind auch mehrere Berichtsdatasets erlaubt. Ein DataSet enthält die Felder der Datenquelle, die im Bericht verwendet werden können sowie ev. berechnete Felder angegeben.

6.1.3.7 Deploy von Datenquellen und Reports

Datenquellen und Reports, die in Visual Studio erstellt wurden, müssen zuerst deployed werden, bevor sie verwendet werden können. Dies kann aus Visual Studio heraus geschehen, indem die jeweilige Komponente rechtsgeklickt und *Deploy* bzw. *Erstellen/Bereitstellen* gewählt wird. Der Ordner, in den die Berichte gespeichert werden, kann mit der Eigenschaft *TargetReportFolder* des SSRS Projekts eingstellt werden. Die Standardeinstellung ist der Pfad *http://localhost/ReportServer.*

6.1.4 Berichtsmanager (Report Manger)

Der Berichtsmanager wird über die URL http://<<Berichtsservername>>/Reports aufgerufen. Im erscheinenden Stammverzeichnis werden bereitgestellte Reports sowie Datenquellen angezeigt. Alternativ zum Deploy aus Visual Studio heraus können hier neue Datenquellen erstellt sowie Reports (RDL-Dateien) hochgeladen werden.

Durch Anklicken der einzelnen Reports in der Liste werden diese angezeigt. Im oberen Bereich besteht dann die Möglichkeit, den Report manuell zu exportieren. Hierbei können die bereits aufgezählten Formate gewählt werden (siehe Kapitel 6.1.3.1).

6.1.4.1 Abonnements

Ist ein Bericht im Berichtsmanager gerade ausgewählt, besteht die Möglichkeit, sogenannte Abonnements zu erstellen, so dass Reports automatisch zur Verfügung gestellt werden können. Dazu muss auf die Registerkarte *Abonnements* und danach auf den Knopf *Neues Abonnement* geklickt werden.

Wie die Reports genau zur Verfügung gestellt werden, hängt von der Konfiguration des Abonnements ab. Die zwei am häufigsten verwendeten Varianten sind das Schreiben von Reports auf eine Dateifreigabe oder das Versenden per E-Mail.

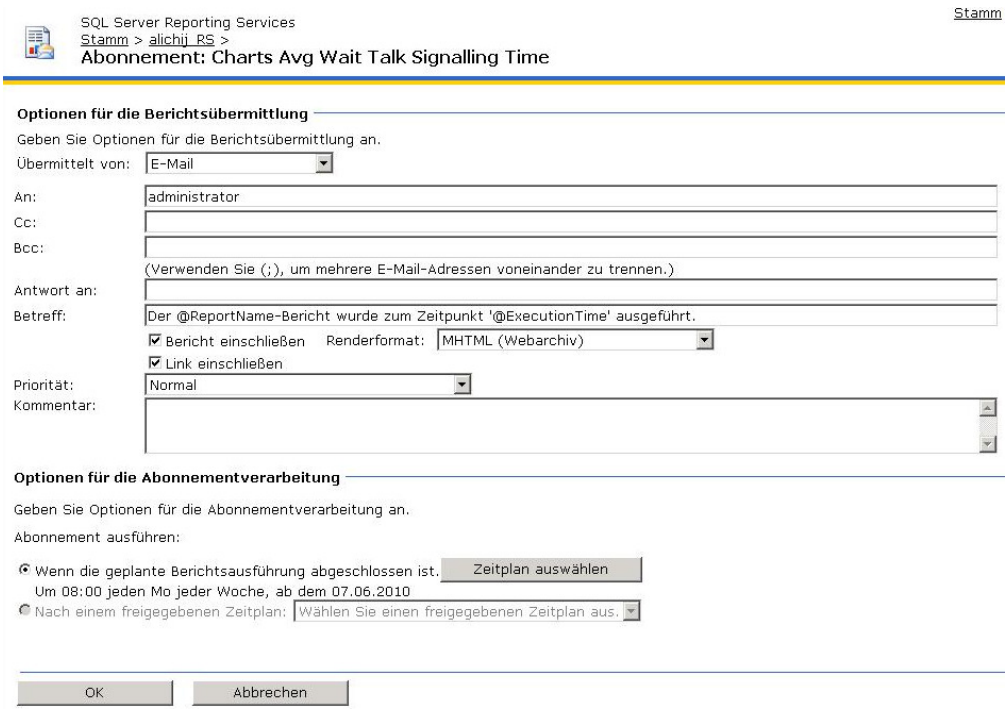


Abbildung 30: E-Mail-Abonnement Konfigurationsmöglichkeiten

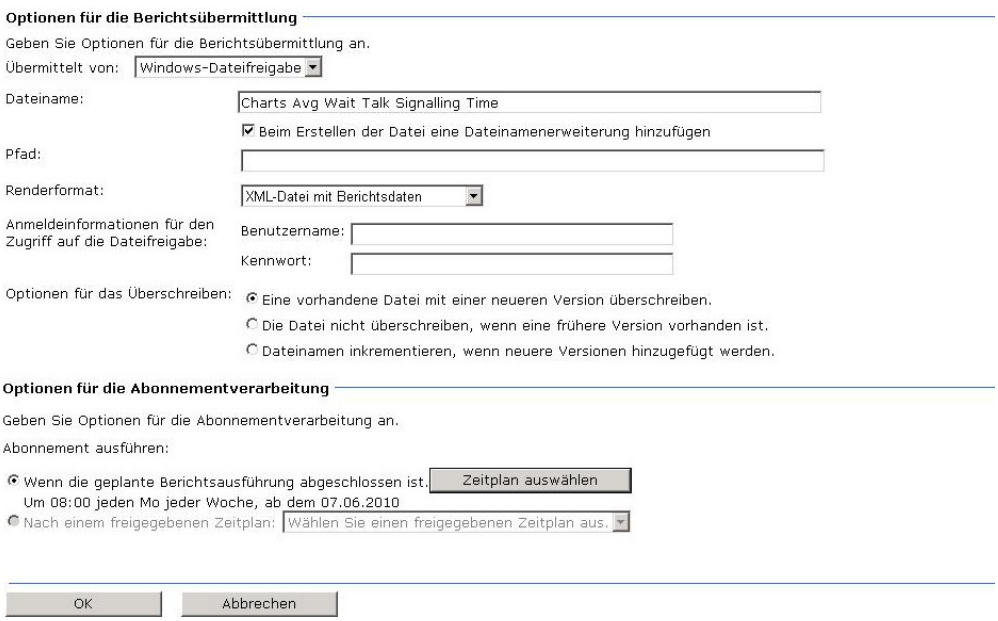


Abbildung 31: Dateifreigabe-Abonnement Konfigurationsmöglichkeiten

6.1.5 SQL Server Agent

Bei dem SQL Server-Agent handelt es sich um einen Windows-Dienst, mit dem Aufträge geplant und einmal oder regelmässig ausgeführt werden können. Nach einer Standardinstallation des SQL Server 2008 wird der SQL Server Agent nicht automatisch gestartet. Um den Startmodus auf *Automatisch* zu stellen oder den Service manuell zu starten, kann der SQL Server Configuration Manager verwendet werden.

Der SQL Server Agent kann dafür verwendet werden, Integration Services Packages regelmässig auszuführen, um beispielsweise die Daten aus der Live Datenbank für das DW aufzubereiten und zu kopieren. Wie Zeitpläne für das Ausführen von Packages erstellt werden, zeigen wir anhand einer Anleitung im Anhang dieses Dokuments.

6.1.6 SQL Server Configuration Manager

Der SQL Server Configuration Manager ist ein Microsoft Management Console Snap-In, um die Services des SQL Servers zu verwalten. Unter *SQL Server Services* ist es möglich, den Startmodus der Services auf *Automatisch* oder *Manuell* zu stellen, sowie die Services manuell zu starten oder zu beenden.

6.1.7 SQL Server Management Studio

SQL Server Management Studio ist ein Werkzeug, um alle Komponenten innerhalb des SQL Servers zu verwalten und administrieren. Ein zentrales Feature des Management Studios ist der Object Explorer, der es dem Benutzer erlaubt, Objekte auf dem Server zu durchsuchen sowie auf ihnen zu agieren.

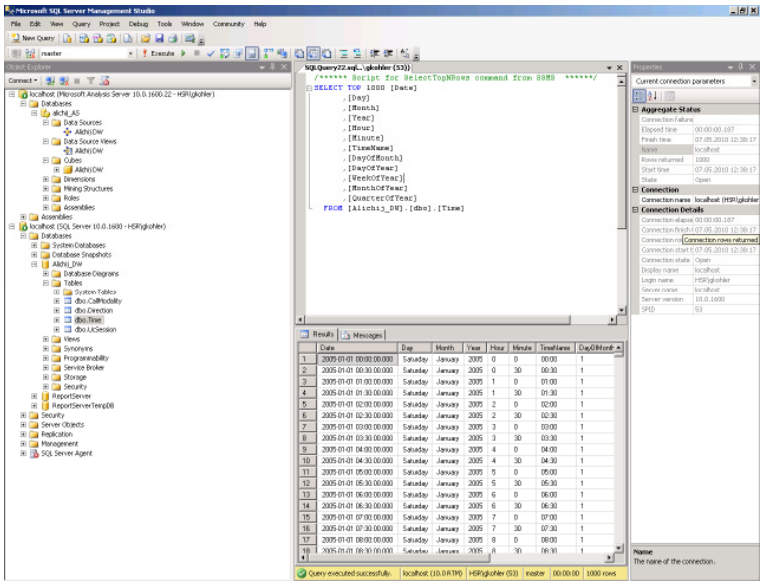


Abbildung 32: SQL Server Management Studio

6.1.8 SQL Server Business Intelligence Developement Studio

Das Business Intelligence Development Studio besteht aus Microsoft Visual Studio 2008 mit speziellen Projekttypen, die nur für SQL Server Business Intelligence verfügbar sind. Dazu gehören die Projekttypen Integration Services Project, Analysis Services Project und Reporting Services Project.

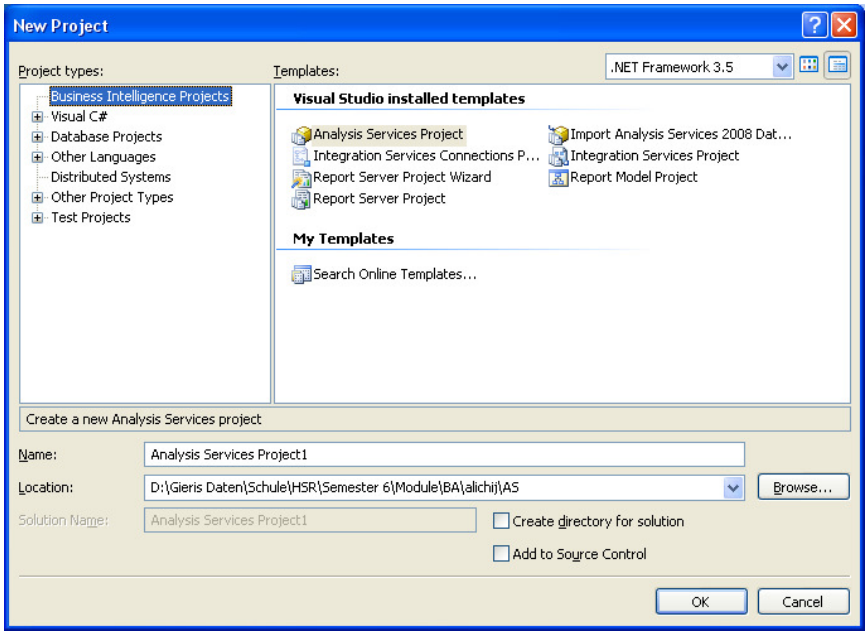


Abbildung 33: Business Intelligence Projects

6.2 ADOMD.NET

ADOMD.NET ist ein Daten-Provider aus dem .NET-Framework von Microsoft, welcher für die Kommunikation mit den SSAS und damit im Gegensatz zum ADO.NET Daten-Provider für mehrdimensionale Daten ausgelegt ist. ADOMD.NET ist der Nachfolger der COM-Komponente ADOMD.

ADOMD.NET benutzt für die Kommunikation mit den Analysis-Services-Datasources das XML for Analysis–Protokoll (XMLA). Die Kommunikation wird mittels XMLA-kompatiblen Simple Object Access Protocol-Objekten (SOAP) entweder über TCP/IP oder über HTTP abgewickelt. Für die Kommunikation über HTTP ist eine Datapump notwendig. Siehe folgendes Kapitel.

Commands können entweder in MDX, Data Mining Extensions (DMX), Analysis Services Scripting Language (ASSL) oder aber mittels limitierter SQL-Syntax abgesetzt werden.

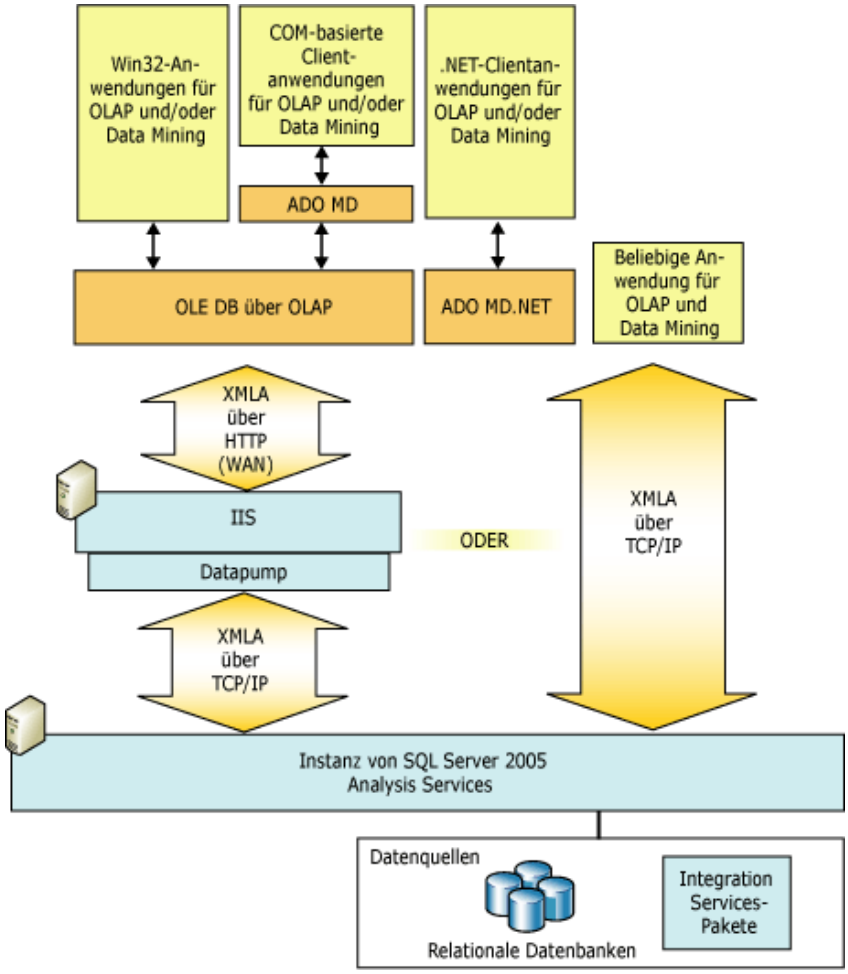


Abbildung 34: ADOMD.NET Übersicht[20]

6.2.1 Datapump

Da die SSAS als Authentifizierungs-Möglichkeit nur die Integrierte Windows-Authentifizierung zulässt, stellen sich diverse Probleme, wenn der Benutzer des ADOMD.NET-Datenproviders z.B. nicht derselben Domain wie der Datenbank-Server mit den Analysis Services angehört. Um dieses Authentifizierungs-Problem zu umgehen, besteht die Möglichkeit, auf dem Datenbank-Server über den Internet Information Services (IIS) die msmdpump.dll zugänglich zu machen. Diese Pump dient als Internet Server API (ISAPI) und pumpt Daten vom Client zu den Analysis Services und zurück. [21] Um Zugriff auf die Daten im SSAS zu erhalten, müssen die Berechtigungen so gesetzt werden, dass der Benutzername, unter welchem die Pump ausgeführt wird, sprich der Service-Account, Zugriffsrechte auf den Cube und die Daten hat. Somit ist die Aufgabe der Authentifizierung der Benutzer an den IIS delegiert worden. Auf dieser Stufe muss im operativen Betrieb eine Lösung für die Authentifizierung des Benutzers gefunden werden.

In der Projektumgebung von LUBI wurde diese Pump durch Andreas Kobler eingerichtet.

[20] Quelle: (Microsoft, Clients (Analysis Services - Mehrdimensionale Daten), 2010)
[21] Anleitung der Einrichtung:
- http://msdn.microsoft.com/en-us/library/cc917711.aspx
- http://bloggingabout.net/blogs/mglaser/archive/2008/08/15/configuring-http-access-to-sql-server-2008-analysis-services-on-microsoft-windows-server-2008.aspx

6.2.2 Rückgabe-Werte

Ein Command, welches über ADOMD.NET abgesetzt wird, wird im XMLA-Format transportiert und die Antwort auch wieder in Selbigen retourniert. Im Client-Code besteht jedoch die Möglichkeit, sich die Antwort in verschiedene Datentypen kapseln zu lassen:

- Cellset
- AdomdDataReader
- XmlReader

Jedes dieser drei Objekte stellt ein Gleichgewicht zwischen Interaktivität und Aufwand her:

- *Interaktivität* bezieht sich auf die Benutzerfreundlichkeit und die Menge an Informationen, die über das Objektmodell zur Verfügung stehen.
- *Aufwand* bezieht sich auf den Datenverkehr, der von einem Objektmodell über die Netzwerkverbindung zum Server erzeugt wird, auf den Speicherplatz, der für das Objektmodell erforderlich ist, und die Geschwindigkeit, mit der das Objektmodell Daten abruft.

Folgende Tabelle stellt die Unterschiede zwischen Interaktivität und Aufwand für jedes Objekt dar:

Objekt	Interaktivität	Aufwand	Dimensionalität
CellSet	Am höchsten	Mäßig hoch; führt zum langsamsten Datenabruf	Wird beibehalten
AdomdDataReader	Mässig	Mässig	Geht verloren
XmlReader	Am niedrigsten	Am niedrigsten; führt zu schnellstem Datenabruf	Wird beibehalten

Tabelle 16: Rückgabe-Objekte ADOMD.NET[22]

6.3 Silverlight

Silverlight ist eine Browser-Erweiterung, welche die Ausführung von Applikationen mit einem Look & Feel und einem Verhalten wie eine Desktotp-Applikation erlaubt. Diese Applikation werden als Rich Internet Applications (RIA) bezeichnet. Silverlight steht mittlerweile in der Version 4 als Plug-In für Windows und MacOS in den Browsern Internet Explorer, Safari und Firefox zur Verfügung.

Wird eine Silverlight-Applikation im Web angesteuert, lädt das Plug-In vom Webserver den Applikations-Code in einem XAP-File auf den Client. Die Kommunikation mit dem Webserver findet typischerweise entweder über HTTP-GET-Request oder über Webservices statt. Silverlight-Applikationen werden in einer kontrollierten Umgebung, der sogenannten Sandbox, ausgeführt und haben nur eingeschränkte Rechte auf dem Client-Computer. Bei Bedarf können Silverlight-Applikationen ab der Version 4 aber explizit lokal installiert werden und erhalten somit zusätzliche Rechte.

6.3.1 Entwicklung

Silverlight-Applikation werden analog zu Windows Presentation-Foundation (WPF)-Applikation entwickelt. Das heisst die GUI-Gestaltung geschieht mittels XAML-Code deklarativ, während für das Implementieren der Logik z.B. VB.NET oder C# verwendet werden kann. Zur Entwicklung werden das neue Visual Studio 2010 und der RC von Microsoft Expression Blend 4 verwendet.

6.3.2 Einschränkungen

Silverlight steht nur eine .NET-Klassenbibliothek zur Verfügung, deren Umfang kleiner als die diejenige des gesamten .NET-Frameworks ist. Dadurch ist z.B. die ADOMD.NET-Bibliothek in einem Silverlight-Projekt nicht verfügbar.

[22] Zitat: (Microsoft, Abrufen von Daten einer analytischen Datenquelle, 2010)

7 Analyse Live Datenbank und bestehendes System

Dieses Kapitel beschreibt die Live Datenbank, welche die auszuwertenden Daten beinhaltet sowie das bereits bestehende System, welches vor bzw. während dieser BA entwickelt worden ist. Es ist hier ganz klar abzugrenzen, dass die Entwicklung der Komponenten, die in diesem Kapitel beschrieben werden, nicht Teil der BA LUBI war. Einige Punkte sollten aber trotzdem angesprochen werden, da sie für das Verständnis wichtig sind.

7.1 Bestehendes System[23]

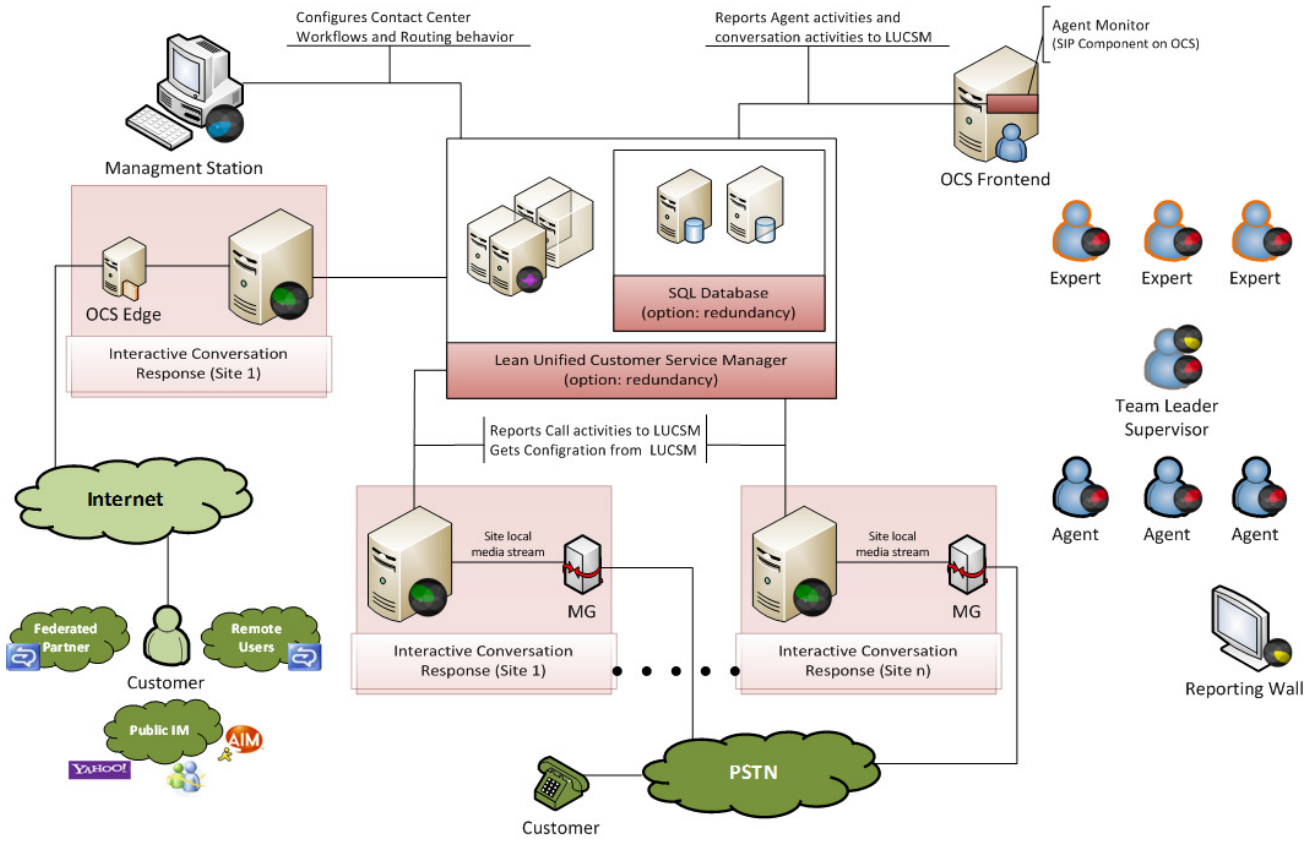


Abbildung 35: Systemübersicht bestehendes System

Abkürzung	Produkt	Farbe	Beschreibung	Ansprechperson
LUCSM	Lean Unified Customer Service Manager	violet	LUCSM ist die Schaltzentrale, wo Statusinformationen über die Kundenberater gehalten und Routing-Entscheidungen getroffen werden. Sie verwaltet ebenfalls die Fähigkeiten der Kundenberater sowie die gesamte Systemkonfiguration.	Andreas Kobler
AM	Activity Monitor	grün	Der AM zeichnet Informationen über Statuswechsel der OCS-User und Anrufsaktivitäten auf und sendet sie dem LUCSM.	Andreas Kobler
ICH*	Interactive Conversation Handler	grün	Die ICH-Komponente stellt einen oder mehrere Service-Eintrittspunkte zur Verfügung (werden adressiert mittels SIP-URI). Sie teilt die Service-Anrufsaktivitäten dem LUCSM mit. Sobald der LUCSM eine	Michael Jakob

[23] Die Inhalte des Kapitels 7.1, namentlich die Abbildung 35: Systemübersicht bestehendes System und die Tabelle 17: Übersicht der Systemkomponenten wurden vom LUCSM Designdokument von Andreas Kobler übernommen

			Routingentscheidung für den Anruf gefällt hat, kann der ICH den Anruf an den richtigen Kundenberater durchstellen. Der ICH ist per B2BUA in jedem Service-Anruf als UAC/UAS involviert und hat somit volle Kontrolle über diese.	
ICR*	Interactive Conversation Response	grün	Die ICR sammelt gemäss einem Call-Workflow Informationen über einen Kunden. Diese werden an den LUCSM übermittelt, worauf Letzterer die Routing-Entscheidung trifft. Um Wartezeiten des Kunden zu überbrücken, spielt der ICR gemäss Workflow Ansagen oder Musik ab. Die Entkopplung des ICR von dem ICH gibt die Möglichkeit, auch ICR-Komponenten von Drittherstellern einzubinden.	Michael Jakob
LUCC	Lean Unified Conversation Context	rot	Über die Conversation Context Komponente wird dem Kundenberater während dem Bedienen eines Service-Anrufes die bereits gesammelten/notierten Informationen dargestellt. Er kann auch weitere Ergänzungen oder Änderungen vornehmen.	Michael Jakob
LUAC	Lean Unified Agent Controller	rot	Der Agent Controller gibt dem Kundenberater eine Übersicht über seinen aktuellen Status und jenem des Contact Centers. Auch Informationen über den Arbeitsplan usw. sind ersichtlich.	Michael Jakob
LUBI	Lean Unified Business Intelligence	gelb	Die Daten aus dem Lifesystem werden regelmässig in ein Datawarehouse übernommen und für historische Reports aufbereitet.	BA, Andreas Kobler
LURW	Reporting Wall	gelb	Die Reporting Wall gibt einem Team die Übersicht über den momentanen Status	Andreas Kobler
LUWF	Lean Unified Workflow	blau	Mit einem Workflow Editor kann der Ablauf eines Service Anrufes einfach und intuitiv implementiert werden.	BA, Michael Jakob

Tabelle 17: Übersicht der Systemkomponenten

*ICH und ICR sind vorerst in einer Komponente vereint

7.2 Beschreibung der Live Datenbank

Anhand des Entitity Relationship Diagramms, wie es zum Ende der Bachelorarbeit aktuell war, beschreibt dieses Unterkapitel die für die Arbeit wichtigen Tabellen. Da das Diagramm in seiner Gesamtheit zu gross ist, werden Ausschnitte daraus gezeigt und erklärt. Das gesamte Diagramm befindet sich zusätzlich als Übersicht im Anhang.

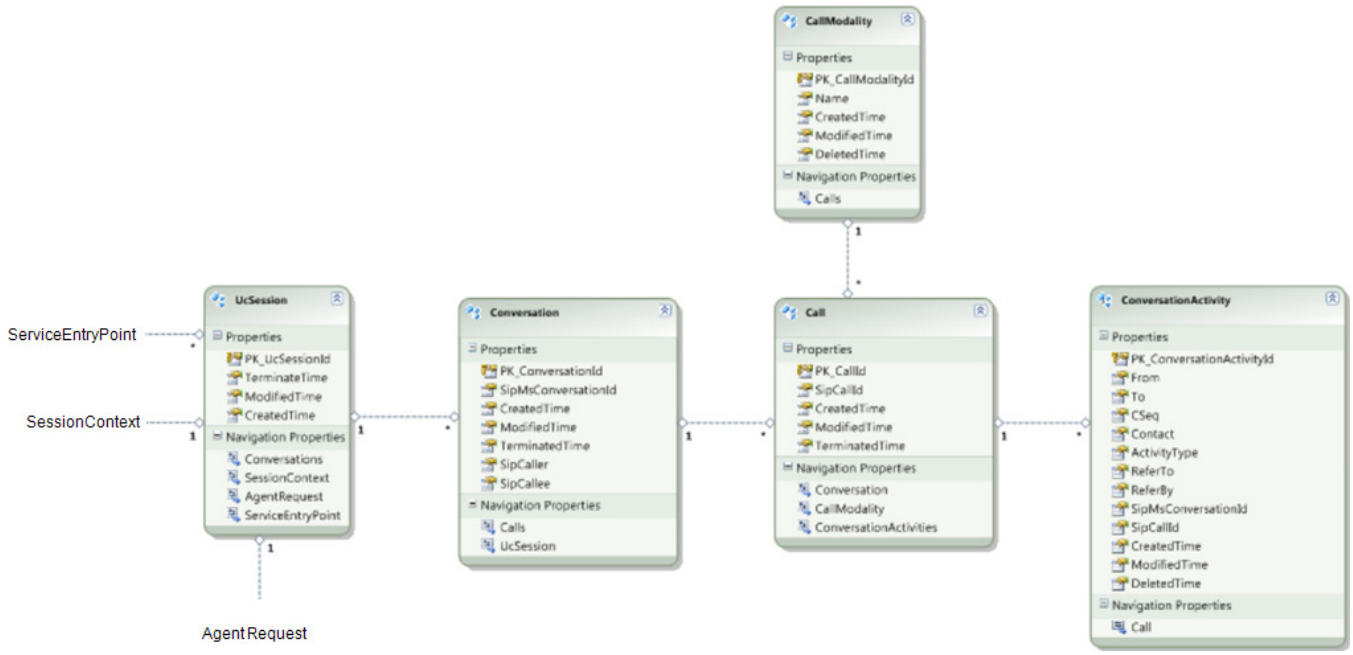


Abbildung 36: ER Diagramm, UcSession und dazugehörige Tabellen

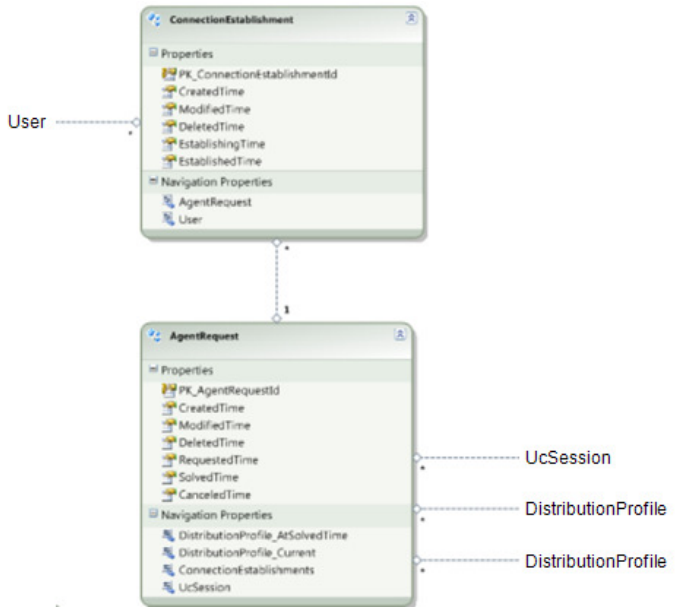


Abbildung 37: ER Diagramm, ConnectionEstablishment und AgentRequest

Tabelle	Erklärung
UcSession	Eine Unified Communication Session besteht aus beliebig vielen Conversations. Eine Service-UcSession beinhaltet alle Conversations, die nötig sind, um den Kunden mit einem oder mehreren Kundenberatern zu verbinden. Eine UcSession kann auch Kundenberater intern miteinander verbinden.
Conversation	Eine Conversation verbindet einen SIP Caller mit einem SIP Callee. Conversations können beliebig viele Calls beinhalten.

Call	Innerhalb einer Conversation werden Calls erstellt. Jedem Call wird eine CallModality zugeordnet, so dass für jede unterschiedliche CallModality ein neuer Call innerhalb der Conversation erstellt wird.
CallModality	Kundenberater und Kunden können über unterschiedliche Modalitäten kommunizieren. Dazu zählen z.B. AudioVideo, InstantMessaging oder DesktopSharing.
ConversationActivity	Diese Tabelle hält alle übertragenen Pakete des SIP Protokolls fest. Im gegenwärtigen Zustand werden für das Reporting keine Informationen aus dieser Tabelle verwendet.
AgentRequest	Die Tabelle wird für Service-UcSessions verwendet. Der AgentRequest wird erstellt, sobald für einen Kunden ein geeigneter Kundenberater anhand von bestimmten Kriterien gesucht wird. Dies geschieht pro Service-UcSession normalerweise einmal, kann aber auch mehrmals geschehen, wenn später weitere Kundenberater hinzugezogen werden. Im Ausnahmefall kann es auch passieren, dass gar kein AgentRequest abgesetzt wird, da der Kunde die Verbindung früher getrennt hat oder die Verbindung aus sonstigen Gründen getrennt wurde.
ConnectionEstablishment	Die Tabelle hält fest, wann ein Kundenberater für einen bestimmten AgentRequest gefunden wurde und wann dieser den Request beantwortet hat (z.B. das Telefon abgenommen hat) Falls der gefundene Kundenberater nicht antwortet, können weitere ConnectionEstablishments erstellt werden.

Tabelle 18: Tabellenbeschreibungen Live Datenbank Teil 1

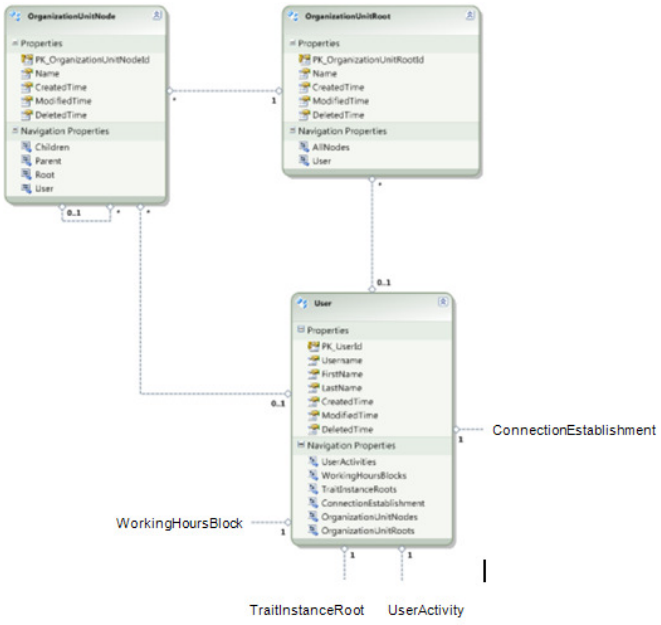


Abbildung 38: ER Diagramm, User und OrganizationUnits

Die Kundenberater bzw. User werden einer OrganizationUnit (OU) zugeordnet. Die OU's können hierarchisch aufgebaut werden, so dass eine Teamhierarchie entsteht.

Tabelle	Erklärung
User	Alle Kundenberater werden in dieser Tabelle abgelegt.
OrganizationUnitNode	Ein OrganizationUnitNode entspricht einem Team oder Segment innerhalb einer Organisation. Die Nodes können verschachtelt werden und daher eine Hierarchie darstellen. Der Node wird durch den Fremdschlüssel *Parent* dem übergeordneten Node zugeordnet. Der ParentNode kann auch Null sein, in diesem Fall ist der über den Fremdschlüssel *Root* zugeordnete Node der ParentNode.
OrganizationUnitRoot	Der OrganizationUnitRoot ist eine Spezialisierung des gewöhnlichen OrganizationUnitNode und stellt im Prinzip nur den obersten aller OU's dar. Jeder OrganizationUnitNode hat einen Verweis auf seinen OrganizationUnitRoot, was den Vorteil hat, dass Nodes, welche unter einem bestimmten Root angeordnet sind, schnell gefunden werden können.

Tabelle 19: Tabellenbeschreibungen Live Datenbank Teil 2

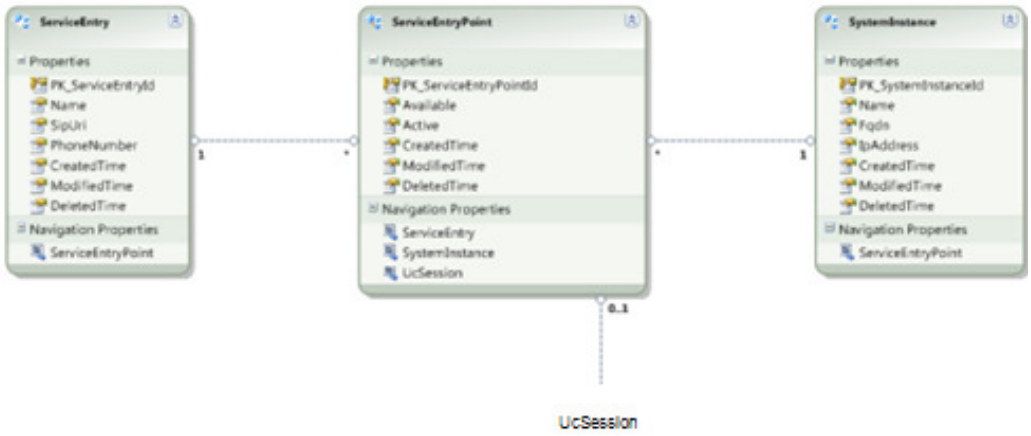


Abbildung 39: ER Diagramm, ServiceEntryPoint und dazugehörige Tabellen

Tabelle	Erklärung
ServiceEntryPoint	Der ServiceEntryPoint stellt einen Bereich in einem Unternehmen dar, in welchem Support durch die Kundenberater zur Verfügung gestellt wird.

Tabelle 20: Tabellenbeschreibungen Live Datenbank Teil 3

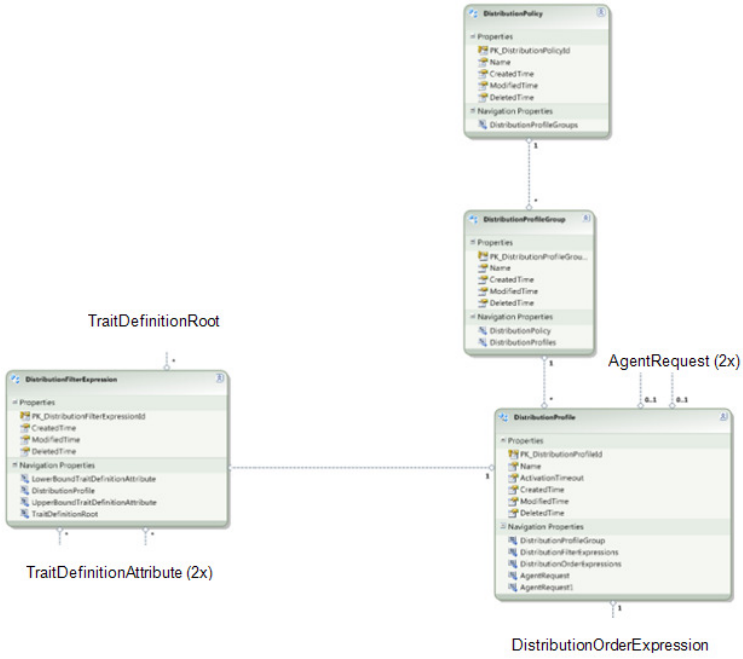


Abbildung 40: ER Diagramm, DistributionProfile und dazugehörige Tabellen

Im bestehenden System wird ein sogenanntes Policy based Call Distribution eingesetzt. Anhand bestimmter Kriterien wird hierbei der beste Kundenberater für einen bestimmten Kunden gesucht. Ein veranschaulichendes Beispiel folgt im folgenden Kapitel.

Tabelle	Erklärung
DistributionPolicy	Eine Policy, die mehrere DistributionProfileGroups enthalten kann, die so gruppiert werden.
DistributionProfileGroup	Eine DistributionProfileGroup wird auf eine bestimmte Kundengruppe abgestimmt. Eine beliebiege Anzahl von DistributionProfiles können ihr zugeordnet werden. Beispiele von DistributionProfileGroups sehen Sie in Abbildung 41: Beispiel Call Distribution Retail Customer (Fokus: Kosten) und Abbildung 42: Beispiel Call Distribution Business Customer (Fokus: Qualität). Jede dieser Abbildungen entspricht einer DistributionProfileGroup, die hier visuell dargestellt wird.
DistributionProfile	Ein DistributionProfile stellt einen Vertriebskreis dar. Ein Kunde wird vorzugsweise immer einem Kundenberater eines primären Vertriebskreises zugeordnet. Vergeht eine gewisse Zeit, ohne dass ein geeigneter Kundenberater gefunden wird, wird die Suche auf einen zweiten, dann auf einen dritten Vertriebskreis ausgeweitet. In Abbildung 41: Beispiel Call Distribution Retail Customer (Fokus: Kosten) und Abbildung 42: Beispiel Call Distribution Business Customer (Fokus: Qualität) entsprechen die einzelnen Farben jeweils einem DistributionProfile bzw. Vertriebskreis.
DistributionFilterExpression	Diese Filter beschreiben für ein DistributionProfile, welche oberen und unteren Schwellwerte für dieses DistributionProfile gelten.

Tabelle 21: Tabellenbeschreibungen Live Datenbank Teil 4

7.3 Policy Based Call Distribution

Bei der Policy Based Call Distribution wird ein geeigneter Kundenberater anhand von Kriterien gesucht, die sich je nach Kunde unterscheiden können. Im Folgenden werden zwei Beispiele gegeben. Ein Kundenberater wird anhand der Kriterien *Expertise*, *Language*, *Distance* und *Responsibility* gesucht.

Das erste Beispiel zeigt eine Distribution Profile Group für einen Retail Customer, das auf möglichst geringe Kosten ausgelegt ist. Wird ein Kundenberater für einen Retail Customer gesucht, versucht das System zuerst einen Kundenberater zu finden, der die Attribute des primären DistributionProfile erfüllt. Konkret werden also alle Kundenberater berücksichtigt, die eine hohe Responsibility (Responsibility: high), eine nahe Entfernung zum Kunden (Distance: close), eine Expertise der Stufe *junior* sowie Sprachkenntnisse der Stufen *native*, *proficient* oder *advanced* haben.

Wenn eine gewisse konfigurierbare Zeit überschritten wurde, ohne dass ein passender Kundenberater im primären Vetriebskreis gefunden wurde, wird die Suche erweitert, so dass auch Kundenberater einbezogen werden, die das sekundäre DistributionProfile erfüllen, in diesem Fall also Kundenberater, die sich weiter entfernt befinden (Distance: *middle* und *far*). Nach einer weiteren Wartezeit one Erfolg werden dann auch Kundenberater in die Suche einbezogen, deren Responsibility nur noch *medium* oder *low* ist. Wie in der Abbildung zu sehen ist, wird für einen Retail Customer beispielsweise nie ein Kundenberater herangezogen, der *expert* ist.

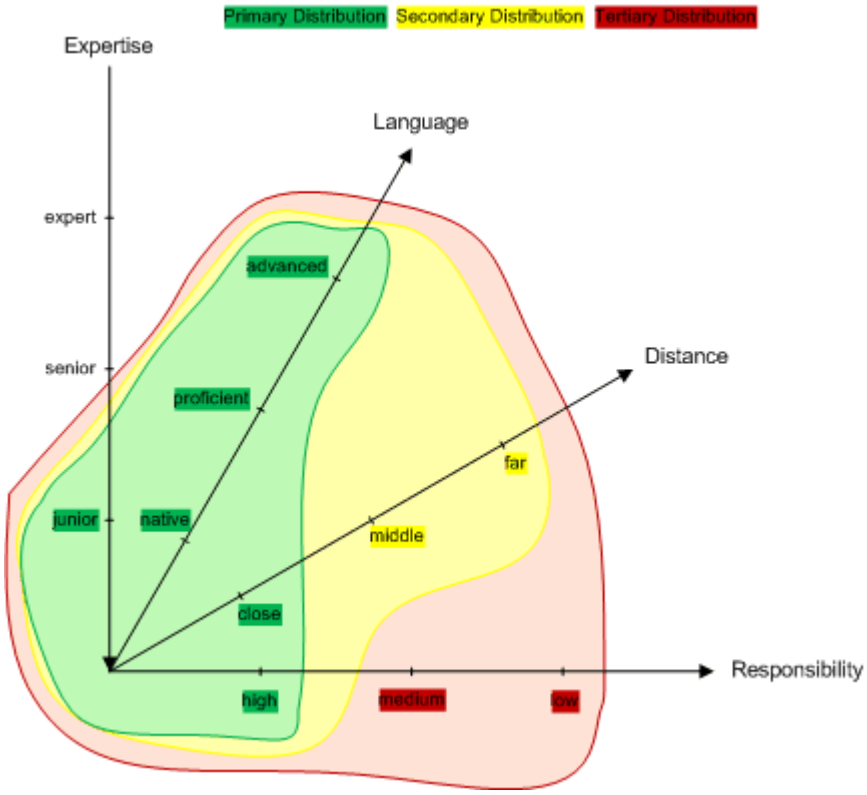


Abbildung 41: Beispiel Call Distribution Retail Customer (Fokus: Kosten)

Das zweite Beispiel zeigt eine DistributionProfileGroup für Business Customer, die mehr Wert auf Qualität legen. Hierbei werden zum Beispiel Experten bevorzugt, Seniors nur noch im sekundären DistributionProfile gesucht und die Juniors gänzlich aus der Suche ausgeschlossen. Auch wird mehr Wert auf eine perfekte Beherrschung der Sprache gelegt. Kundenberater mit den Sprachkenntnissen *proficient* und *advanced* werden nur noch im zweiten bzw. dritten Vertriebskreis für die Suche berücksichtigt.

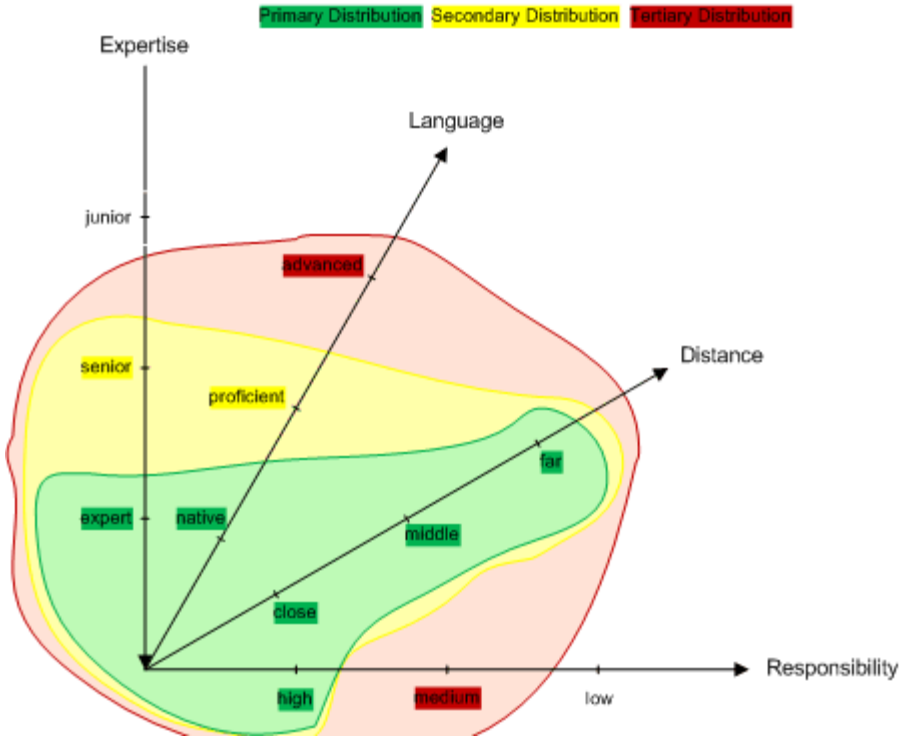


Abbildung 42: Beispiel Call Distribution Business Customer (Fokus: Qualität)

7.4 UcSession Modell[24]

Als erstes wird auf die interne (Internal) UcSession eingegangen, da diese einfacher aufgebaut ist wie die restlichen UcSession-Typen. Bei einer internen UcSession kontaktiert ein Kundenberater innerhalb des Contact Centers einen anderen Kundenberater, ebenfalls innerhalb des Contact Centers.

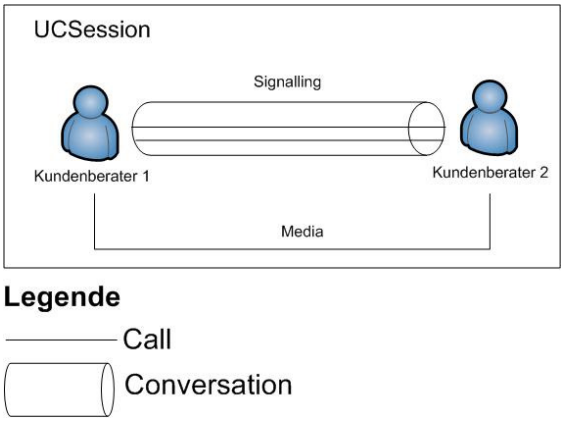


Abbildung 43: Interne UcSession mit zwei Kundenberatern

Bei einer internen UcSession zwischen zwei Kundenberatern wird eine einfache Conversation, die einen Call beinhaltet, zwischen beiden Kundenberatern erstellt. Wenn weitere Modalitäten (IM, AudioVideo, Desktop-Sharing) hinzukommen, werden neue Calls innerhalb der Conversation erstellt. Die Conversation wird nur zur Signalisierung (SIP) eingesetzt, die eigentlichen Daten (Media) laufen direkt zwischen den beiden Kundenberatern, wie in obiger Abbildung zu sehen ist.

[24] Weitere Informationen: http://msdn.microsoft.com/en-us/library/dd253342%28v=office.13%29.aspx

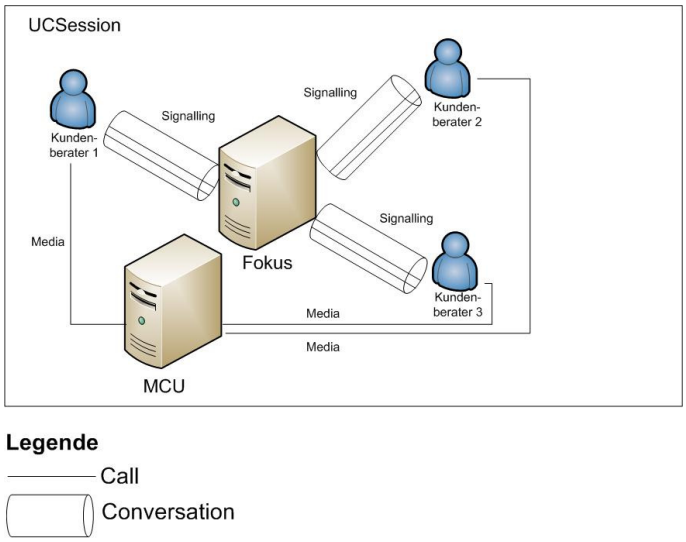


Abbildung 44: Interne UcSession mit drei Agenten (Conference)

Es besteht nun die Möglichkeit, dass einer der beiden Kundenberater einen weiteren Kundenberater der UcSession hinzufügt, so dass eine Konferenz entsteht. In diesem Fall wird ein sogenannter Fokus eingefügt, über welche die Signalisierung läuft. Die Übertragung der Daten erfolgt über die ebenfalls neu hinzugekommene MCU (Multipoint Conversation Unit)

7.4.1 Eingehende (Inbound) und ausgehende (Outbound) UcSessions

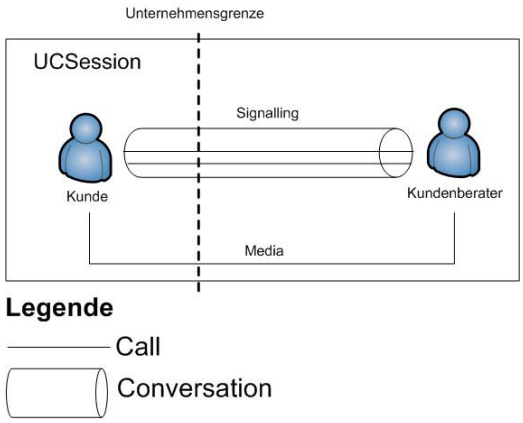


Abbildung 45: Eingehende und ausgehende UcSessions

Ausgehende UcSessions gehen von einem Kundenberater aus, der beispielsweise einen Kunden ausserhalb des Unternehmens kontaktiert. Eingehende UcSessions werden von ausserhalb des Unternehmens ausgelöst, wenn ein Kundenberater auf direktem Weg (Direktwahlnummer) kontaktiert wird. Der Umweg über die Agentensuche wie bei einer Service-UcSession wird hier also nicht gemacht. Das Modell für eingehende und ausgehende UcSessions sieht gleich aus. Der einzige Unterschied besteht darin, dass ausgehende UcSessions von innen und eingehende UcSessions von aussen ausgelöst werden. Die Signalisierungen laufen also in umgekehrter Richtung.

7.4.2 Service-UcSessions

Die Darstellungen zeigen unter Anderem die zwei Komponenten ICH und MCU. Der ICH (Interactive Conversation Handler) ist dafür zuständig, anhand von Kriterien einen für den Kunden möglichst passenden Kundenberater zu finden (siehe Kapitel 7.3) Die MCU (Multipoint Conversation Unit) stellt das Zentrum einer Conference im Backend dar. Diese Conference beinhaltet alle Kundenberater, die während einer UcSession der Unterhaltung hinzugefügt werden.

Achtung: Wichtig ist, dass diese Conference nicht verwechselt wird mit dem Zustand *Conference*, in der sich eine UcSession befinden kann (siehe Kapitel 8.4). Vom Zustand *Conference* einer UcSession sprechen wir, wenn mehr als zwei Kommunikationspartner miteinander in Kontakt sind. Diese Information ist für das Reporting relevant. Hier jedoch wird

angestrebt, dass immer eine Conference erstellt wird, auch wenn nur ein Kunde mit einem einzigen Kundenberater kommuniziert. Die MCU ist das Zentrum dieser Conference und hält diese zusammen.

Bei der Modellbeschreibung der Service-UcSessions unterscheiden wir zwischen dem Modell, wie es am Ende der BA aktuell war und dem angestrebten Modell, welches noch umgesetzt werden soll.

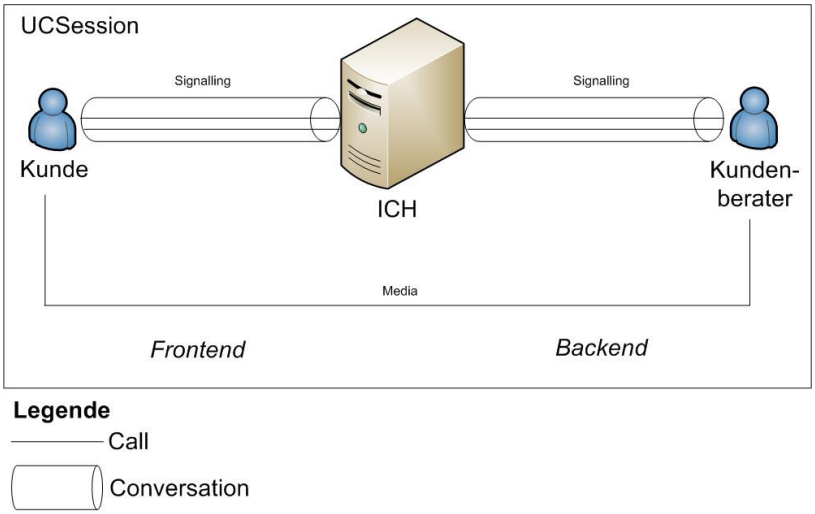


Abbildung 46: aktuelles Modell für Service-UcSession, Stand Ende BA

Im Modell unterscheiden wir zwischen Frontend und Backend. Als Frontend wird der Teil bezeichnet, der in den Abbildungen links des ICH positioniert ist. Dementsprechend ist der rechte Teil das Backend.

Im aktuellen Modell (Abbildung 46: aktuelles Modell für Service-UcSession, Stand Ende BA) wird eine Conference über die MCU erst aufgebaut, wenn ein weiterer Kundenberater hinzugezogen wird. Solange der Kunde mit dem Kundenberater kommuniziert und somit nur zwei Kommunikationspartner vorhanden sind, wird darauf noch verzichtet. Es ist jedoch vorgesehen (Abbildung 47: angestrebtes Modell), diese Conference gleich zu Beginn zu erstellen, wenn die Verbindung zum ersten Kundenberater aufgebaut wird. Vorteile dieses Ansatzes sind grössere Flexibilitäten und Möglichkeiten beim Handling der Service-UcSessions. Ein Beispiel dafür ist das einfachere Voice-Recording durch Hinzufügen eines speziellen Recorder-Endpunkts in die Conference.

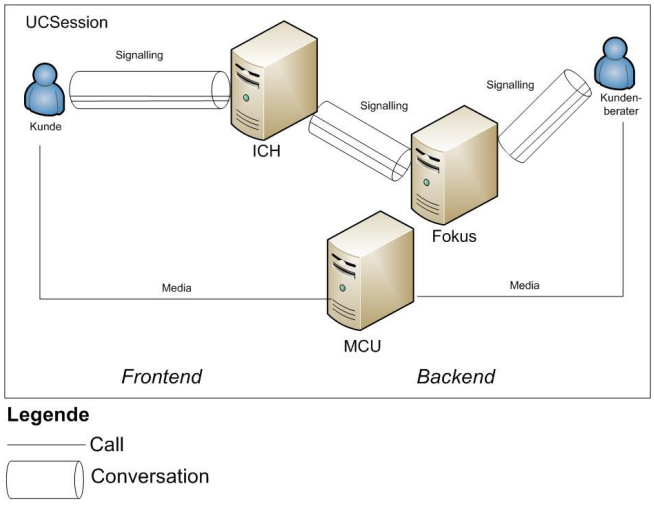


Abbildung 47: angestrebtes Modell für Service-UcSession

8 Architektur und Design LUBI

8.1 Architektur

Die folgende Abbildung zeigt die Komponenten, welche die Reportinglösung realisieren, wobei die Live Datenbank nicht zu LUBI gehört, sondern nur als Datenbasis für die Reports dient. Mittels der Integration Services werden die benötigten Daten aus der Live Datenbank in das Data Warehouse übernommen. Darauf aufbauend wird dann durch die Analysis Services ein OLAP Cube bereitgestellt. Die Reporting Services erlauben dann das Generieren von Reports, indem durch MDX-Abfragen die Daten vom Cube geholt werden. Weiter wurde ein Silverlight-Frontend realisiert, welche die Daten ebenfalls durch MDX-Abfragen auf dem Cube erhält und die erhaltenen Daten grafisch darstellt.

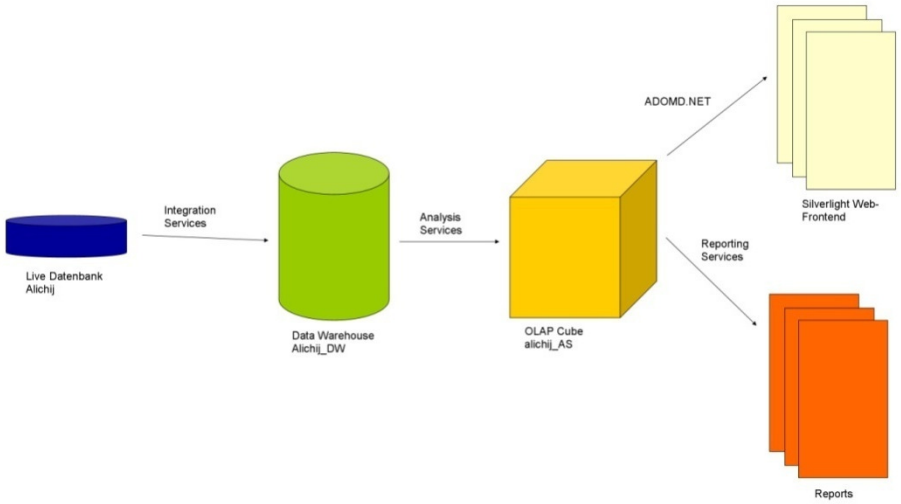


Abbildung 48: Architektur

8.2 Design Vorgehen

Beim Design des Starschemas war die Vorgehensweise an die in (Ross & Kimball, 2002) beschriebene Methode *Four Step Dimensional Design Process* angelehnt. Diese Methode besteht aus den folgenden vier Schritten.

Definition des Business Prozesses:

„Select the business process to model“

Der erste Schritt besteht darin, den Business Prozess zu wählen, der modelliert werden soll.

Festlegen der Granularität:

„Declare the grain of he business process“

Im zweiten Schritt wird die Granularität festgelegt. Damit wird genau spezifiziert, was eine individuelle Faktentabelle repräsentiert. Die Granularität beantwortet die Frage „Wie beschreiben wir eine einzige Zeile in der Faktentabelle?“

Wahl der Dimensionen:

„Choose the dimensions that apply to each fact table row“

Im dritten Schritt werden die Dimensionen gewählt, die auf jede Zeile in der Faktentabelle angewendet werden können. Die Dimensionen beantworten die Frage „Wie beschreiben wir die Daten, die vom Business Prozess resultieren?“

Identifikation der Fakten:

„Identify the numeric facts that will populate each fact table row“

Der letzte Schritt besteht darin, die Fakten in der Faktentabelle (Measures) zu bestimmen. Diese Fakten geben die Antwort auf die Frage „Was messen wir?“

Es lässt sich sagen, dass der Prozess von (Ross & Kimball, 2002) teilweise wiederholt und Korrekturen gemacht werden mussten. Die folgenden Abschnitte beschreiben nur das Endergebnis.

8.3 Identifikation der Business Prozesse

Folgende Business Prozesse wurden identifiziert:

- Service-UcSession auf ServiceEntryPoint
- Internal UcSessions (von Kundenberater zu Kundenberater)
- Outbound UcSessions (vom Kundenberater nach aussen)
- Inbound UcSessions (von aussen zum Kundenberater)

8.4 Service-UcSession auf ServiceEntryPoint

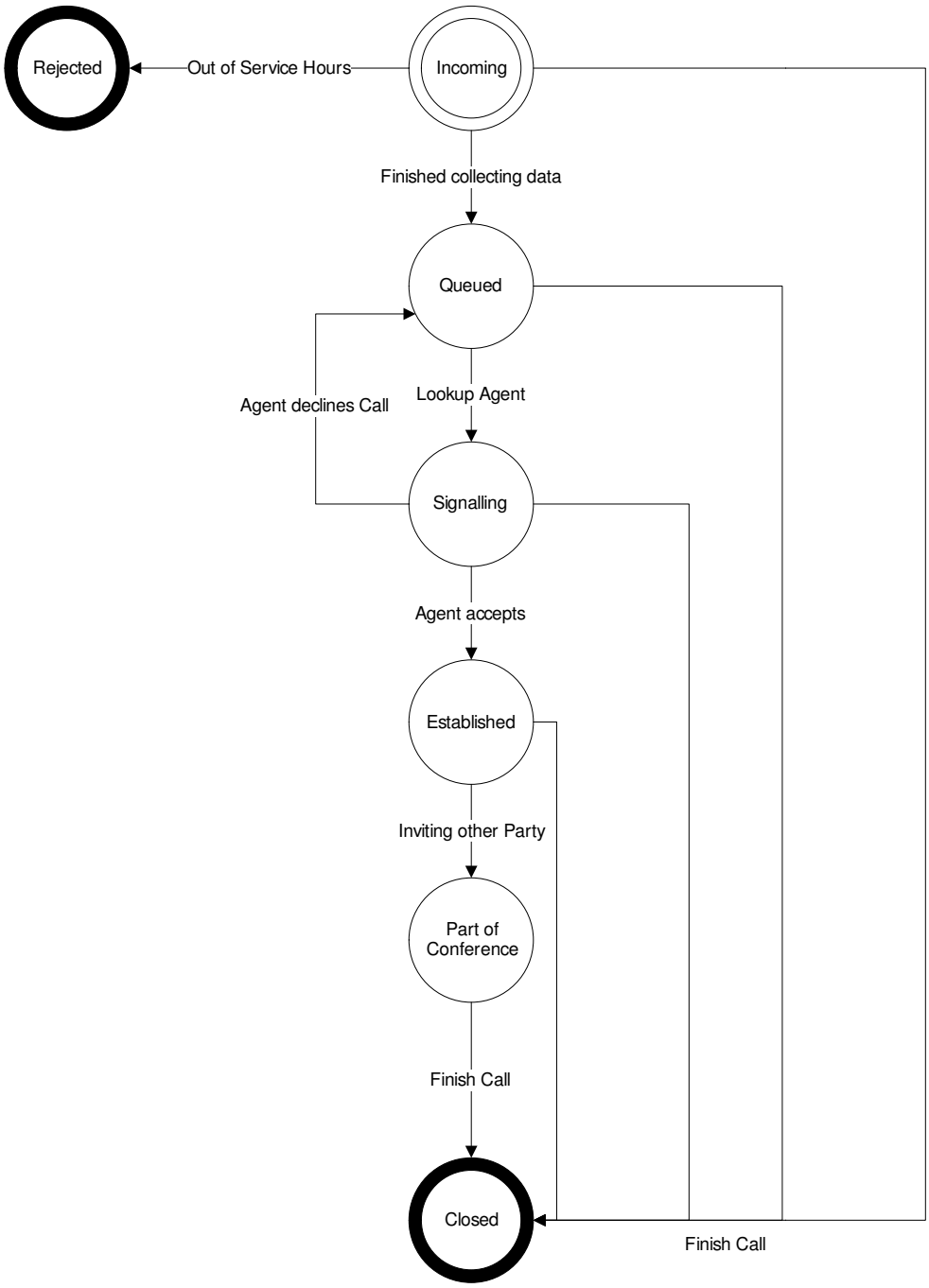


Abbildung 49: Business Prozess *Service-UcSession auf ServiceEntryPoint*

Ein wichtiger Business Prozess ist die Service-UcSession, die von ausserhalb des Contact Centers durch einen Kunden ausgelöst wird, wenn er auf einen ServiceEntryPoint verbindet. Hierbei sind die Möglichkeiten nicht nur auf das Telefon beschränkt. Mögliche Modalitäten sind z.B. AudioVideo, InstantMessaging oder DesktopSharing. Eine Service-UcSession befindet sich während ihrem Lebenszyklus in verschiedenen Zuständen, wie es in der obigen Abbildung dargestellt ist. Die Zustände sind folgendermassen definiert:

Zustand	Beschreibung
Incoming	Der Kunde wählt verschiedene Parameter wie zum Beispiel Service oder Sprache aus.
Queued	Der Kunde wartet in der Warteschlange. Ein geeigneter Kundenberater wird gesucht. Dem Kunden wird Musik vorgespielt.
Signalling	Ein geeigneter Kundenberater wurde gefunden. Dieser wird angefragt und muss nur noch bestätigen, dass er den Kunden bedienen will.
Established	Die Verbindung ist fertig aufgebaut. Der Kundenberater und der Kunde können miteinander kommunizieren.
Conference	Die UcSession besteht aus mehreren Calls, die zusammen eine Konferenz ergeben. Von einer Konferenz wird gesprochen, wenn mehr als 2 Personen miteinander kommunizieren.
Closed	Der Kundenberater oder der Kunde hat die Service-UcSession beendet.
Rejected	Wenn ausserhalb der Öffnungszeiten des Contact Centers versucht wird, eine Service-UcSession auf einen ServiceEntryPoint durchzuführen, erhält diese den Zustand *Rejected*.

Tabelle 22: Zustände einer Service-UcSession

8.4.1 Granularität

Grundsätzlich kamen zwei Granularitäten in Frage. Einerseits waren beinahe alle Anforderungen zu erfüllen, wenn die Granularität *UcSession* gewählt worden wäre, denn User (Kundenberater), Distribution und ServiceEntryPoint verändern sich über die gesamte UcSession nicht. Eine Einschränkung war jedoch, dass nur eine Initialmodalität pro UcSession abgelegt werden konnte. Da aber über eine UcSession mehrere unterschiedliche Modalitäten vorkommen können und die Auswertung der Modalitäten sowie deren Zeiten eine Anforderung war, eignete sich dieser Ansatz nicht. Um die Modalitäten, deren Zeiten und Längen zu erfassen, wäre die Granularität *Call* die richtige Entscheidung gewesen. Da für die meisten Dimensionen die Granularität *UcSession* geeigneter war, wurde entschieden, zwei Granularitäten zu wählen und diese miteinander zu verknüpfen.

Da ein User (Kundenberater) einer OU (Organizational Unit) zugeordnet ist und diese Zuordnung für das Reporting wichtig ist, wurde entschieden, eine Dimension *OU* einzuführen, die von der Dimension *User* verwendet wird. Somit wird die Dimension User zur Faktendimension (siehe Kapitel 8.6.1) und es entsteht eben auch die Granularität *User*.

8.4.2 Dimensionen

Anhand der Anforderungen in Form der User Stories wurden die folgenden Dimensionen ausgewählt:

Dimension	Beschreibung
Distribution	z.B. primärer, sekundärer, tertiärer Vertriebskreis (siehe Kapitel 0)
CallModality	z.B. AudioVideo, InstantMessaging, DesktopSharing
ServiceEntryPoint	z.B. Drucker, Monitore, usw.
User	Contact Center Kundenberater/Agent
TimeOfDay	Zeit, in der eine UcSession bzw. ein Call begann (gerundet auf eine Sekunde)
Date	Datum, in der eine UcSession begann
OU	Die OU (Organization Unit) beschreibt ein Team oder Segment einer Unternehmung

Tabelle 23: Dimensionen Business Prozess *Service-UcSession auf ServiceEntryPoint*

8.4.3 Fakten

Die Measures der Faktendimensionstabelle *FactDim_ServiceUcSession* sind die Zeitdauern, in welchen sich die Service-UcSession in den verschiedenen Zuständen befunden hat sowie weitere berechnete Zeitdauern.

- TimeInIncomingState
- TimeInQueuedState
- TimeInSignallingState
- TimeInEstablishedState
- TimeInConferenceState
- HandledInTime
- AbandonedInTime
- WaitTime
- TalkTime
- UcSessionDuration
- IsOutOfService

Das einzige Measure der Faktentabelle *Fact_ServiceCall* ist:

- CallDuration

Die Faktendimensionstabelle *FactDim_User* hat keine Measures.

8.4.4 Starschema

Die folgende Abbildung zeigt das finale Starschema des Business Prozesses *Service-UcSession auf ServiceEntryPoint*:

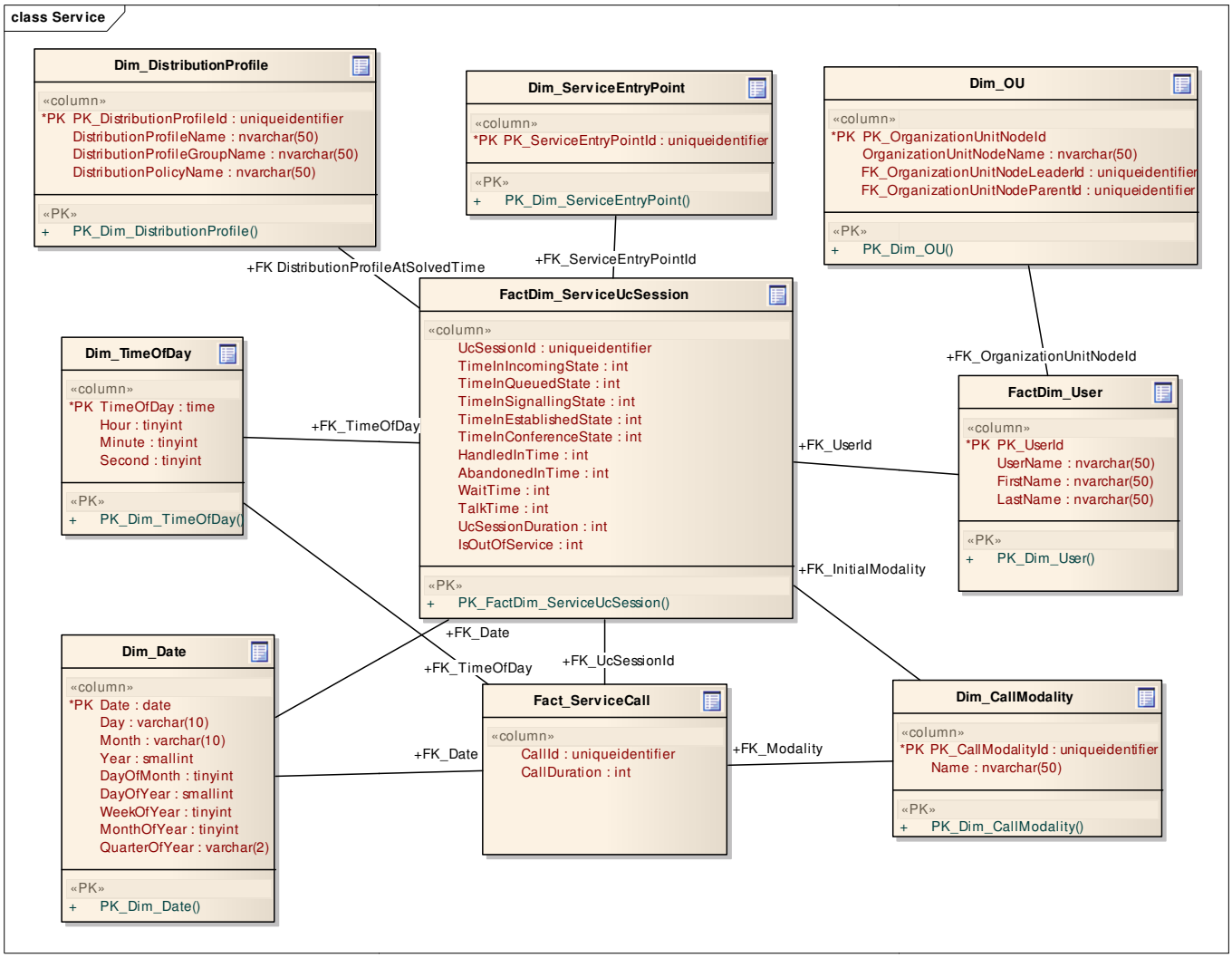


Abbildung 50: Star Schema des Business Prozesses *Service-UcSession auf ServiceEntryPoint*

8.5 Business Prozesse Outbound/Inbound/Internal UcSessions

Diese drei Business Prozesse konnten aufgrund ihrer Ähnlichkeit zusammengefasst werden in ein einziges Starschema. Inbound UcSessions sind keine Service-UcSessions. Sie durchlaufen nicht den Prozess des Suchens eines Kundenberaters, sondern dieser wird direkt angewählt. Kundenberater können auch ausgehende (Outbound) sowie interne (internal) UcSessions aufbauen. Die Unterscheidung dieser drei Business Prozesse wird durch die neue Dimension *Direction* ermöglicht, welche drei Werte beinhaltet, für jeden Business Prozess einen.

8.5.1 Granularität

Die Granularität wurde vom Business Prozess *Service-UcSession auf ServiceEntryPoint* übernommen. Es handelt sich um die Granularitäten *UcSession* und *Call* sowie die Granularität *User*.

8.5.2 Fakten

Das einzige Measure der Faktendimensionstabelle *FactDim_Outbound_Inbound_Internal_UcSession* ist:

- UcSessionDuration

Das einzige Measure der Faktentabelle *Fact_Outbound_inbound_Internal_Call* ist:

- CallDuration

Die Faktendimensionstabelle *FactDim_User* hat keine Measures.

8.5.3 Dimensionen

Dimension	Beschreibung
Modality	z.B. AudioVideo, InstantMessaging, DesktopSharing
User	Contact Center Kundenberater/Agent
Direction	Outbound, Inbound oder Internal
TimeOfDay	Zeit, in der eine UcSession bzw. ein Call begann (gerundet auf eine Sekunde)
Date	Datum, in der eine UcSession begann
OU	Die OU (Organization Unit) beschreibt ein Team oder Segment einer Unternehmung

Abbildung 51: Dimensionen Business Prozesse *Outbound/Inbound/Internal UcSessions*

8.5.4 Starschema

Die folgende Abbildung zeigt das finale Starschema der Business Prozesse *Outbound/Inbound/Internal UcSessions*:

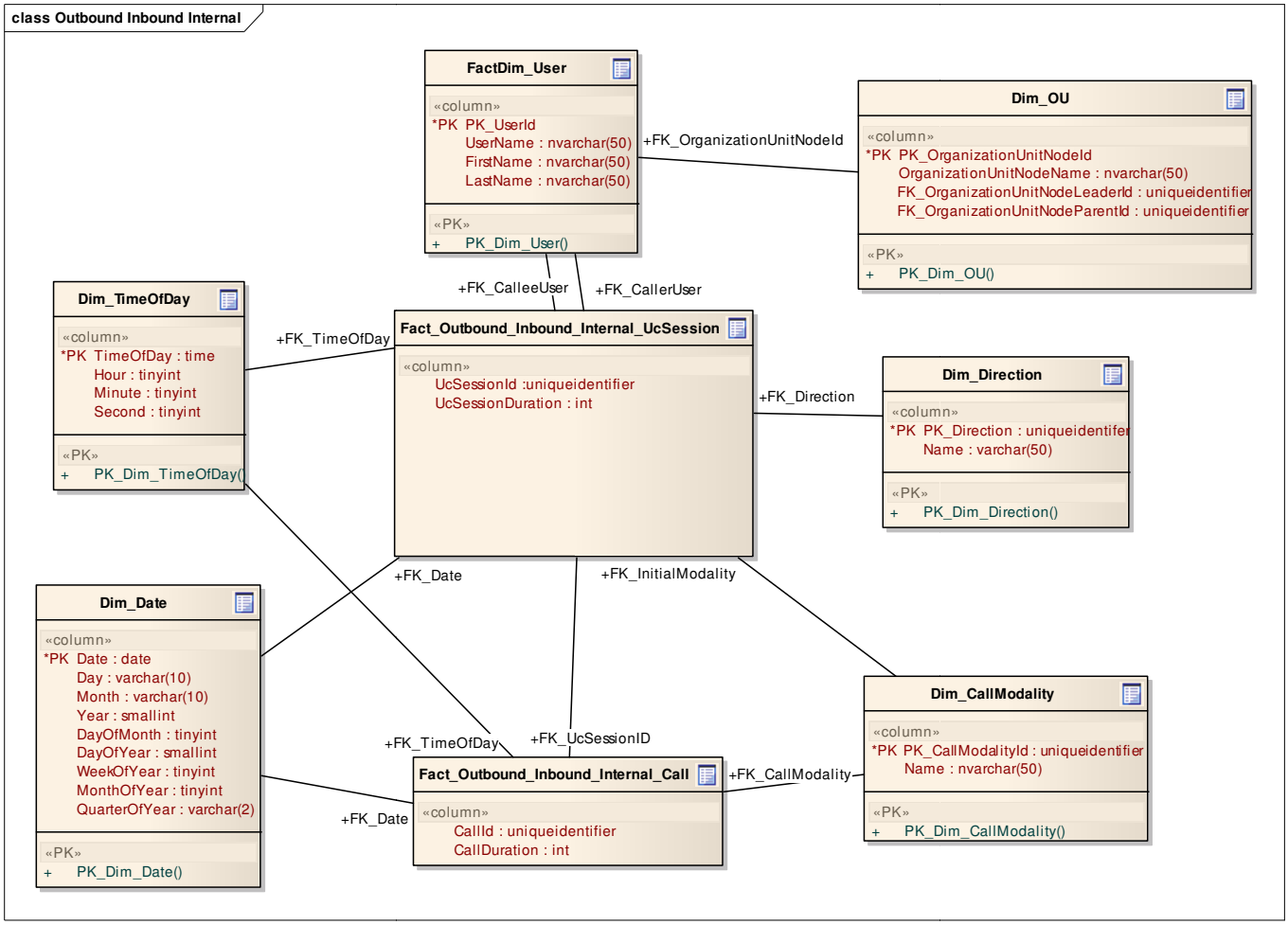


Abbildung 52: Star Schema der Business Prozesse *Outbound/Inbound/Internal UcSessions*

8.6 Erläuterungen zu Designentscheiden

In diesem Abschnitt wollen wir noch auf einige Besonderheiten eingehen und ein paar unserer Gedanken beim Design erläutern.

8.6.1 Faktendimensionen (Fact Dimensions)

Ein bekanntes Pattern erlaubt es, sogenannte Faktendimensionen zu erstellen. Unter anderem wird es im bereits erwähnten (Ross & Kimball, 2002) beschrieben. Auch auf diversen Internetseiten findet man Informationen.

Die sogenannten Faktendimensionen sind, wie der Name schon sagt, eine Mischung aus Faktentabellen und Dimensionstabellen. In beiden unserer Starschemas wird die UcSession als Faktendimensionstabelle gewählt, da sie einerseits in Bezug auf die Faktentabellen *Fact_ServiceCall* bzw. *Fact_Outbound_Inbound_Internal_Call* als Dimension funktioniert. Es können also alle Calls ausgewählt werden, die zu einer bestimmten UcSession gehören. Andererseits hat die Faktendimensionstabelle UcSession auch eigene Measures und verwendet Dimensionen. Auch die Tabelle *User* ist eine Faktendimensionstabelle.

8.6.2 Mehrfache Verwendung der Dimensionen

Wie in beiden Starschemas erkennbar ist, können Dimensionen von verschiedenen Faktentabellen verwendet werden. In unserem Beispiel verwenden die Faktentabellen für die UcSessions und die Calls teilweise die gleichen Dimensionen.

8.6.3 Optimierung der Zeitdimension

Für die Erfüllung der Anforderungen ist es nötig, einem Call bzw. einer UcSession eine Startzeit zuzuordnen, die innerhalb von 5 Jahren liegen kann und auf eine Sekunde genau sein muss. Für jede dieser Zeiten bräuchte es genau einen Datensatz in der Zeitdimensionstabelle. Für die gestellten Anforderungen wären dies 60*60*24*365*5=157‘680‘000 Datensätze. Diese unvorstellbar grosse Menge an Datensätzen würde die Performance von Abfragen auf dem resultieren Cube massiv verschlechtern.
Ein ebenfalls in Ross & Kimball beschriebenes Pattern schlägt 2 Dimensionstabellen vor. Eine speichert das Datum (Dim_Date) und eine weitere die Uhrzeit (Dim_TimeOfDay). Durch diese Vorgehensweise wird die Anzahl der Datensätze massiv verringert. In der Tabelle *Dim_Date* ergeben sich 365*5=1825 und in der Tabelle *Dim_TimeOfDay* 60*60*24=86400 Datensätze. Insgesamt sind also nur 86400+1825 = 88225 Datensätze nötig, was nur 0.00056% von ursprünglich 157‘680‘000 entspricht.

9 Architektur und Design LUBIWebFrontend

Die Applikation LUBIWebFrontend wurde als eigene Solution im Microsoft Visual Studio 2010 erstellt. Sie ist in insgesamt 17 Projekte gegliedert, wovon jedes einzelne Projekt nach der Kompilierung eine eigenständige DLL-Datei ergibt. Nachfolgend werden diese einzelnen Projekte, resp. ihre DLL-Dateien, Komponenten genannt. Zusammen werden die erstellten und referenzierten DLL's schlussendlich in XAP-Files gepackt, welches die downloadbare Silverlight-Applikation darstellt. Die einzelnen Komponenten sind selber teilweise in logische Schichten gegliedert.

9.1 Grobübersicht

Im folgenden eine kurze und grobe Übersicht über die einzelnen Komponenten:

Komponente	Beschreibung
LUBIWebFrontend	Hauptprojekt, beinhaltet Shell und Bootstrapper, somit oberster UI-Container
LUBIWebFrontend.Web	Stellt einerseits den Webservice für den Zugriff auf die SSAS zur Verfügung und beinhaltet zudem die Webpages, in welche die Silverlight-Applikation eingebettet wird.
LUBIWebFrontend.Common	Enthält die ViewModel-Interfaces, die verwendeten Events und EventArgumente sowie den ServiceWrapper für den Zugriff auf den Webservice
LUBIWebFrontend.TabModule	Modul, welches als Container für das Dashboard und die Charts dient.
LUBIWebFrontend. HeaderModule	UserControl, welches das UI und das ViewModel des Headers enthält
LUBIWebFrontend.Dashboard	Enthält das UI und das ViewModel des Dashboards
LUBIWebFrontend.[ChartName]ChartModule	Enthalten die einzelnen Detail-Ansichten der Charts, insgesamt elf Stück.

Tabelle 24: Komponenten LUBIWebFrontend

9.2 Physische Architektur

Im Folgenden ist die Verteilung der Komponenten und benötigten Webservices aufgezeichnet:

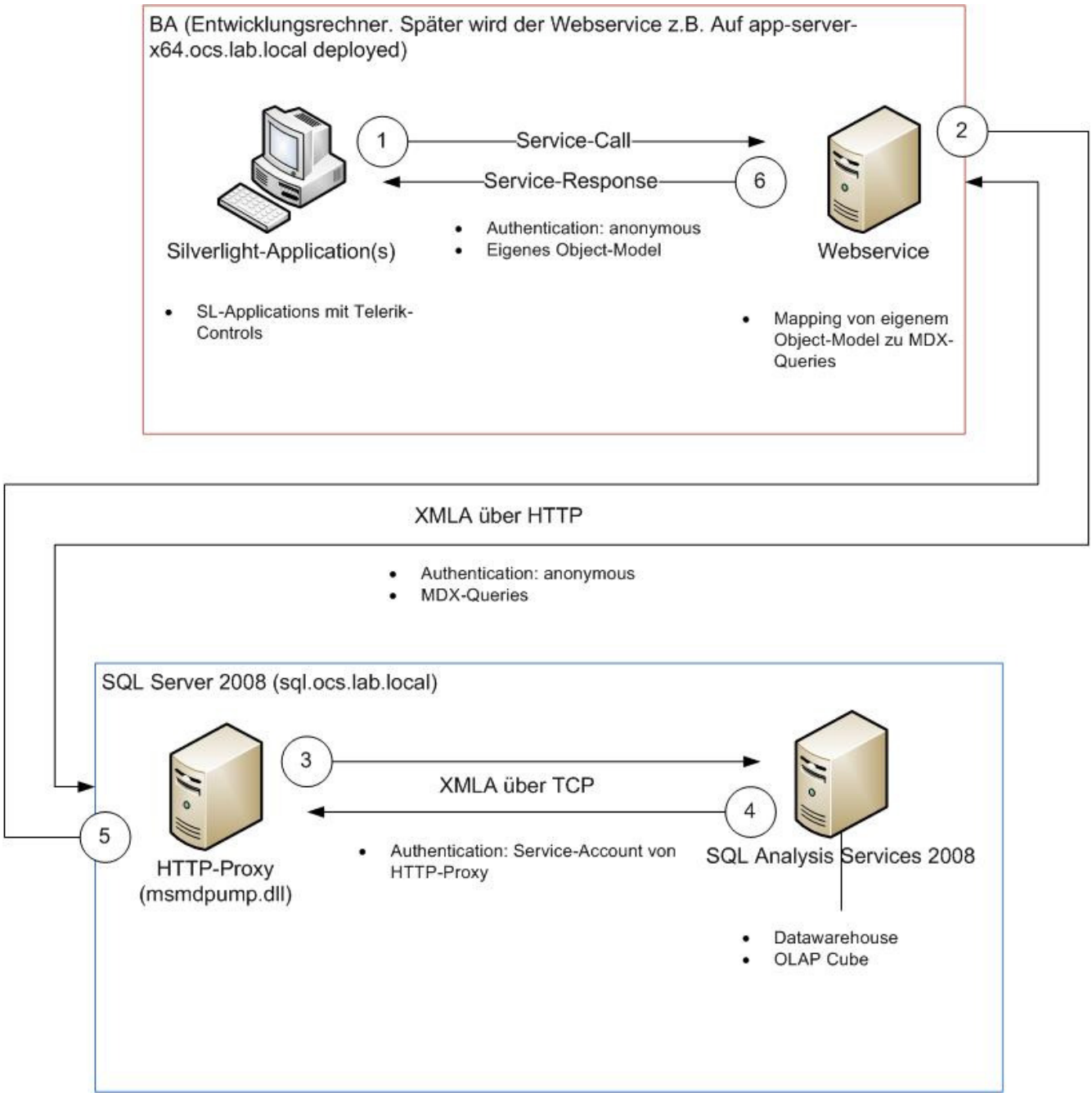


Abbildung 53: Physische Architektur[25]

Der Nutzen resp. der Zweck des zum Einsatz kommenden HTTP-Proxy vor der eigentlichen SSAS-Instanz wurde in Kapitel 6.2.1 angesprochen. Der eigene Web-Service welcher vor die Analysis-Services geschaltet wird, ist nötig, da in Silverlight-Applikationen der Einsatz von ADOMD.NET nicht möglich ist (Kapitel 6.3.2).

9.3 Logische Architektur

Die gesamte WebFrontend-Applikation ist im übergeordneten Namespace *LUBIWebFrontend* abgelegt. Die einzelnen Komponenten haben jeweils ihren eigenen Namespace. Bei den Modulen, welche für das UI zum Einsatz kommen, kommen eigene Namespaces für das ViewModel und für die View zum Zug. In der Common-Komponente existiert ein Namespace für den ServiceAccess, in welchem der Zugriff auf den Webservice gewrappt wird sowie ein Namespace für die verwendeten Events und Event-Argumente. Ein weiterer Namespace beinhaltet die ViewModel-Interfaces der einzelnen ViewModels der

[25] Nach einer Vorlage von Andreas Kobler

Modules. Die Web-Komponente, welche den Webservice für den Zugriff auf die SSAS beinhaltet, enhält die Namespaces *ErrorHandling* und *ProblemDomain*.

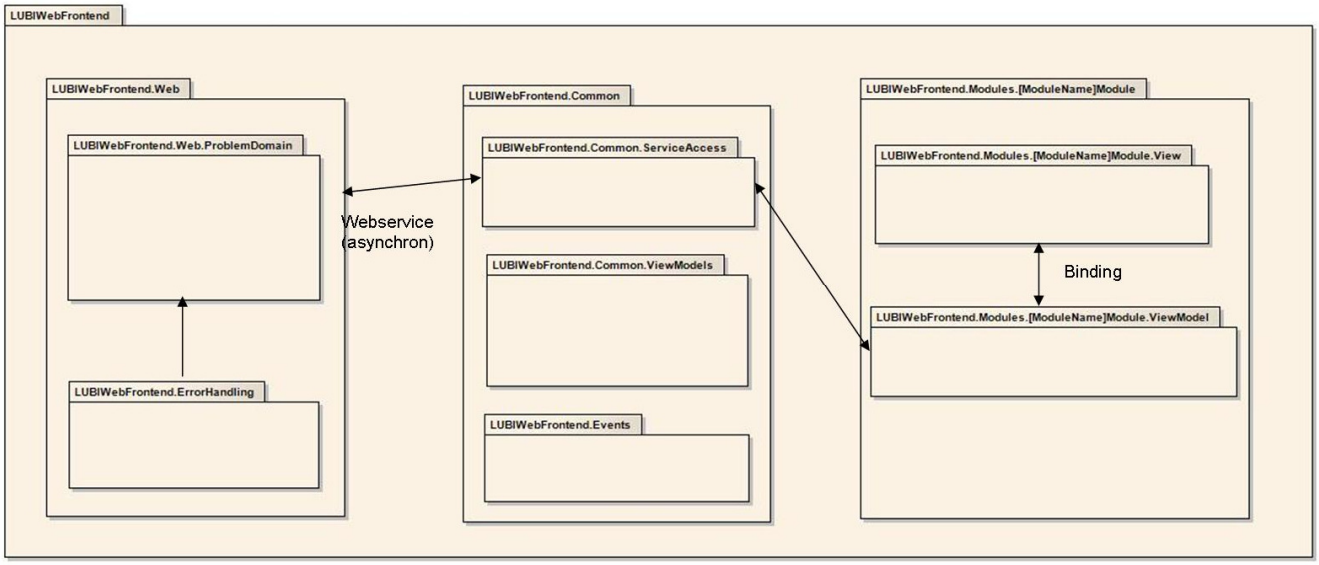


Abbildung 54: Logische Architektur

9.4 Einsatz von Prism/Model-View-ViewModel-Pattern[26]

Wie bereits mehrfach erwähnt, wurde das UI des WebFrontends stark modularisiert. Das soll später die Entwicklung weiterer Komponenten nach spezifischen, zurzeit noch unbekannten Kundenbedürfnissen vereinfachen. Zudem können so einzelne Teile des UI's einfach und schnell angepasst oder ausgetauscht werden. Damit kann die Lebensdauer der Applikation vergrössert werden, weil sich bei ändernden Bedürfnissen problemlos weitere Module in die bestehende Applikation einpassen lassen.

Um diese starke Modularisierung zu erreichen, kommt Prism, früher bekannt unter dem Namen Composite Application Guidance for WPF and Silverlight aus dem Hause Microsoft zum Einsatz. Die einzelnen Module einer Prism-Applikation sind lose gekoppelt. Damit können sie unabhängig von anderen Modulen weiterentwickelt werden. Trotzdem arbeiten alle Module zusammen und bilden die Gesamtapplikation.

Ein Modul einer Prism-Applikation besteht typischerweise immer aus den gleichen Teilen:

Teil	Beschreibung
Module	Verwaltungsklasse, lädt Unity-Container, EventAggregator und RegionManager
ViewModel	Beinhaltet jegliche Daten, Commands für die Anzeige im UI und die Benutzerinteraktion
View	Deklarative Beschreibung des UI (XAML). Daten und Commands werden mittels Binding aus dem ViewModel in die View eingebunden

Tabelle 25: Teile Prism-Module

Prism setzt verschiedene Composite Application Patterns um, darunter z.B. das CompositeView-, das Command- und das Event Aggregator-Pattern.[27]

[26] Quelle: (Microsoft, Technical Concepts, 2010)
[27] Weitere Informationen: http://msdn.microsoft.com/en-us/library/dd458924.aspx

9.4.1 Region-Design

Die LUBIWebFrontend-Applikation wurde in folgende Regions aufgeteilt, in welche anschliessend einzelne Module geladen werden:

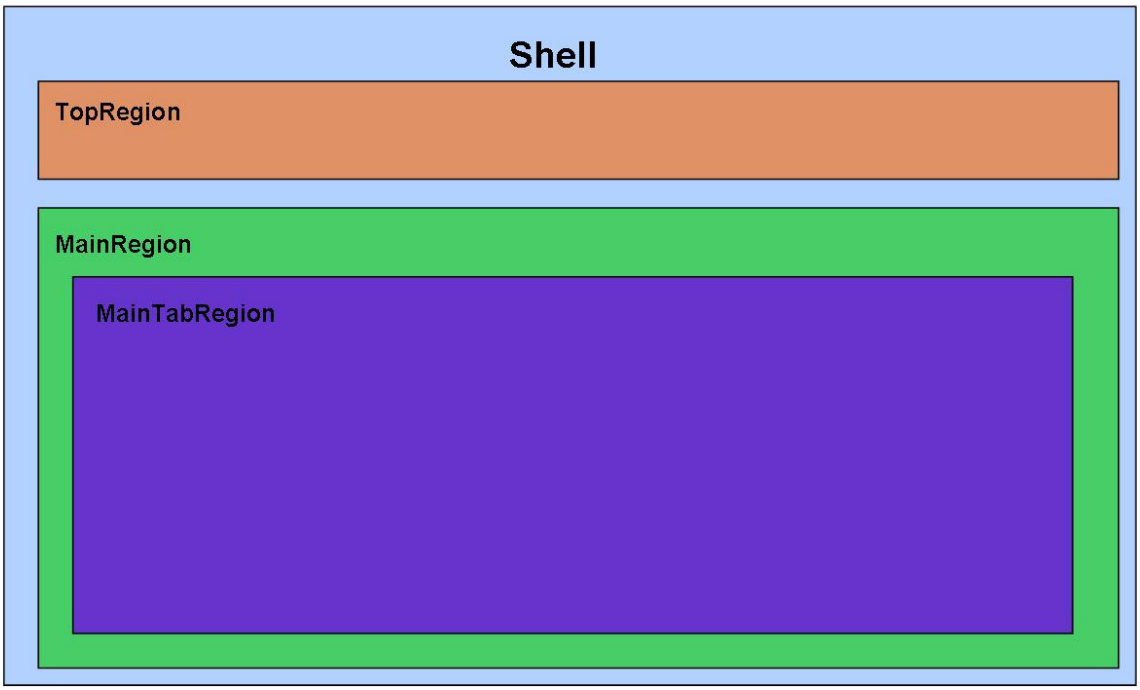


Abbildung 55: Aufteilung der Regions

In die TopRegion wird das HeaderModule geladen. Es beinhaltet das LUBI-Logo und in einer späteren Version die Buttons für die Konfiguration der Applikation, den Aufruf der Hilfefunktion und die Statusanzeige des Logins.

In die MainRegion wird das *TabModule* geladen. Das TabModule mit der MainTabRegion dient als Container für das *DashboardModule* und die *ChartModule*. Für jedes Module in der MainTabRegion wird ein eigenes TabPanel in der MainRegion erstellt. Siehe auch Kapitel 11.2.1.1

9.4.2 Event Handling

Für das EventHandling zwischen einzelnen Modulen wird der EventAggregator verwendet. Dabei werden Publisher und Subscriber durch den EventAggregator entkoppelt, damit sie sich unabhängig voneinander entwickeln können. Im LUBIWebFrontend wird der EventAggregator dazu verwendet, die Detail-Ansichten der Charts vom Dashboard-Modul her einzublenden.

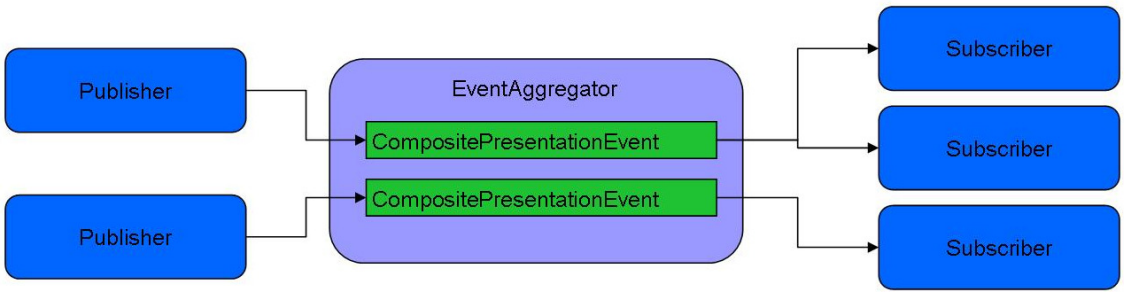


Abbildung 56: Event Handling mit EventAggregator

Die konkreten CompositePresentationEvents, welche für das dynamische Erweitern der View mit Detailansichten von Charts als Tabs ausgelöst werden, sind in der Komponente LUBIWebFrontend.Common im Namespace *LUBIWebFrontend.Common.Events* implementiert.

9.5 Komponenten

9.5.1 LUBIWebFrontend

Die LUBIWebFrontend-Komponente beinhaltet die Klasse *App* und ist somit Einstiegspunkt in die Silverlight-Applikation. Die Klasse *Shell* ist der Hauptcontainer der UI-Komponenten gemäss dem Vorgehen in Prism. Die Klasse *Bootstrapper* ist für die Initialisierung der Prism-spezifischen Aspekte der Applikation zuständig. Er konfiguriert zuerst den Container, anschliessend das Region Adapter Mapping, kreiert die Shell und initialisiert schlussendlich die Module. Der *Bootstrapper* wird aus der Application_Startup()-Methode der *App*-Klasse initialisiert.

Die Klasse *RadTabControlRegionAdapter* wird dafür verwendet, das Telerik-Control RadTab (Container des RadTabPanels) für die Verwendung als Region-Host benutzbar zu machen.

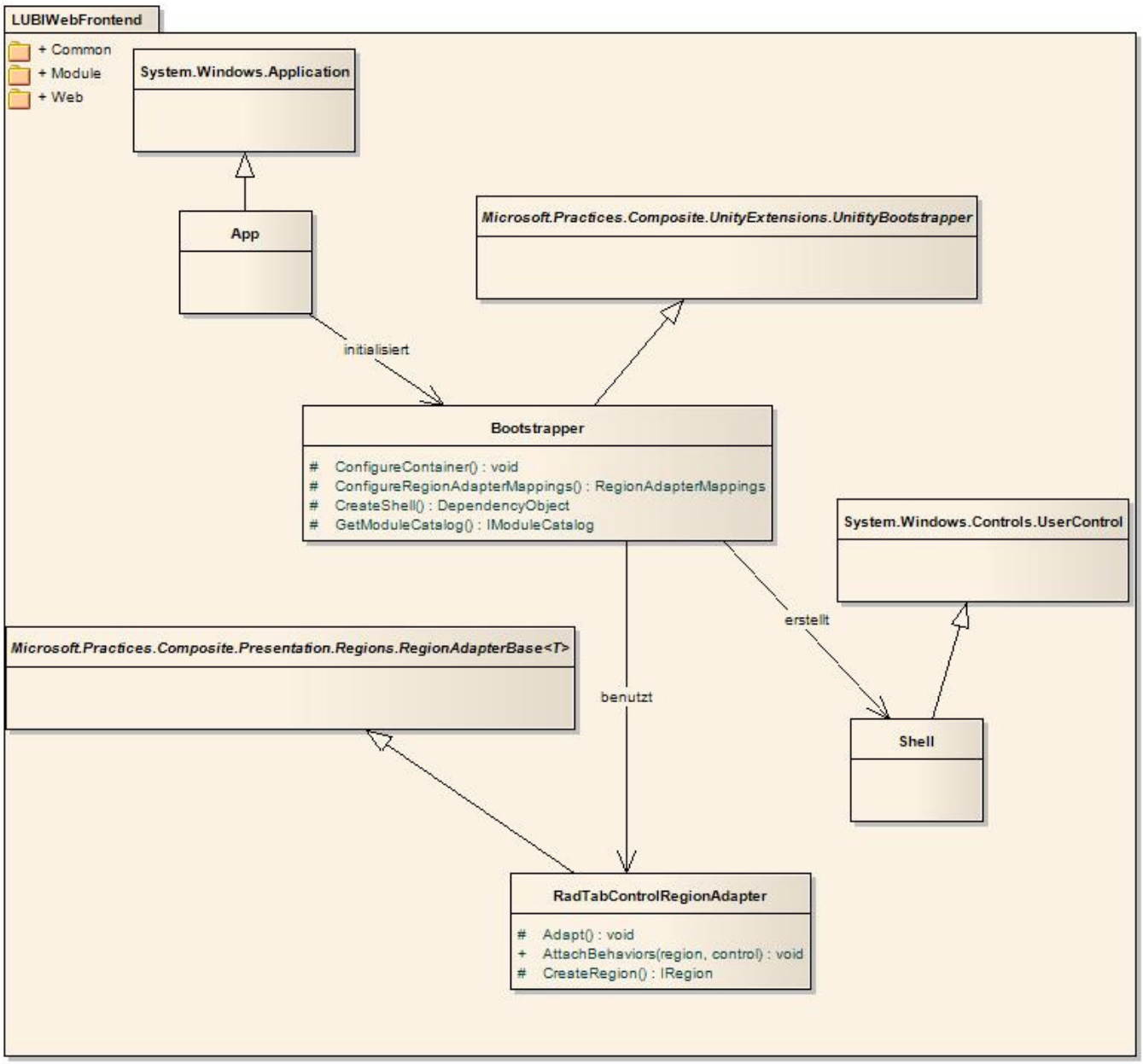


Abbildung 57: Klassendiagramm Namespace LUBIWebFrontend

9.5.2 LUBIWebFrontend.Common

In der Common-Komponente werden die Marker-Interfaces für die ViewModels der einzelnen Module, der Wrapper inklusive Interface für die Kapselung der automatisch erstellten Webservice-Access-Klasse und die verwendeten Events abgelegt. Die Interfaces der ViewModels und des ServiceWrappers dienen als Marker-Interfaces beim Laden der Prism-Applikation.

Folgende Namespaces werden im LUBIWebFrontend.Common aufgespannt:

9.5.2.1 Events

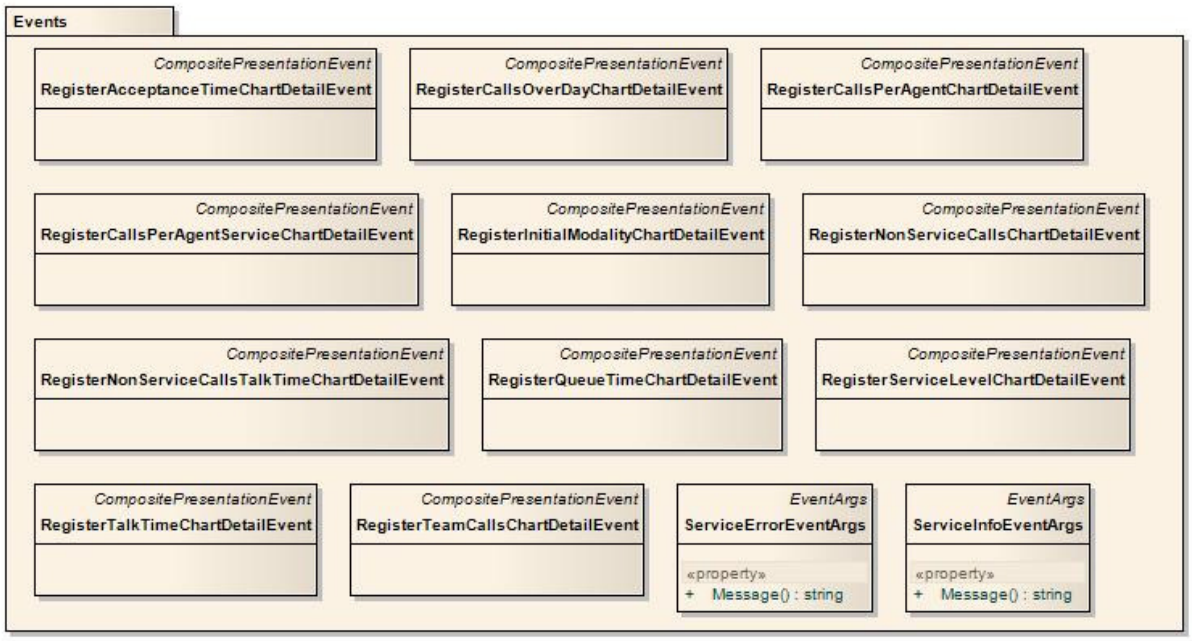


Abbildung 58: Klassendiagramm Namespace LUBIWebFrontend.Common.Events

9.5.2.2 ViewModels

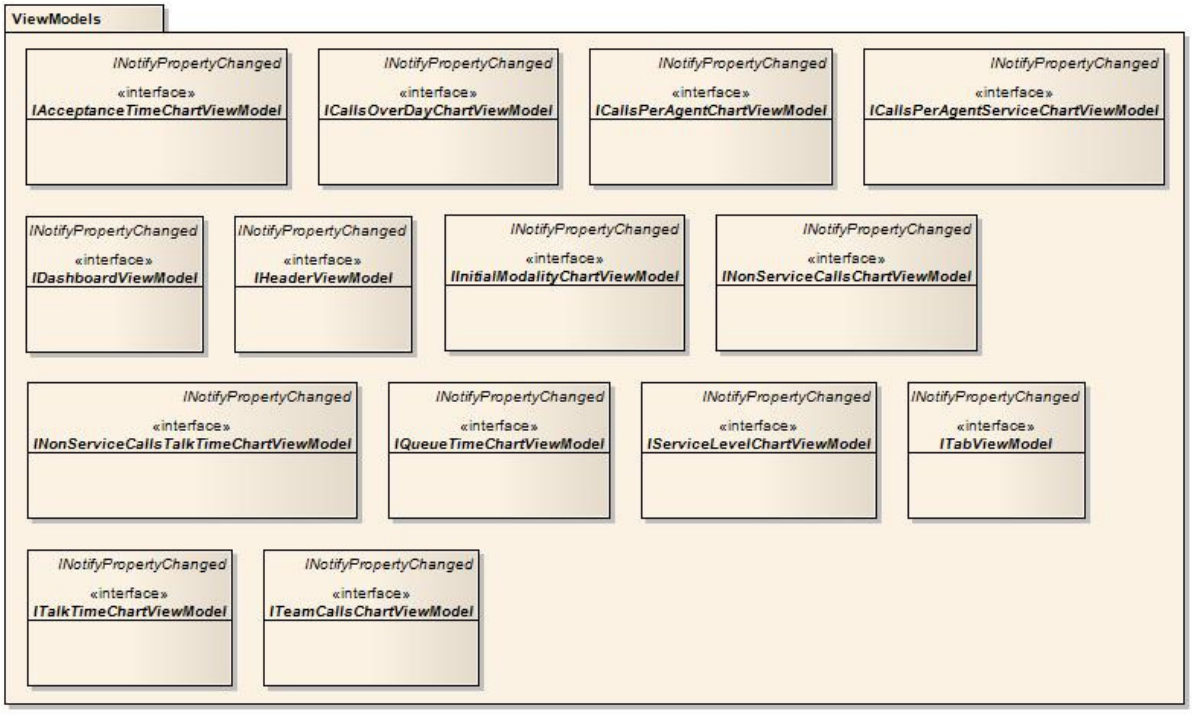


Abbildung 59: Klassendiagramm Namespace LUBIWebFrontend.Common.ViewModels

9.5.2.3 ServiceAccess

Abbildung 60: Klassendiagramm Namespace LUBIWebFrontend.Common.ServiceAccess

9.5.3 LUBIWebFrontend.Web

Die Komponente LUBIWebFrontend.Web beinhaltet neben der HTML und ASPX-Seite für das Hosting der Silverlight-Applikation den AnalysisServicesQueryService. Dieser Webservice liefert die Daten für die Darstellung der Charts. Der Service sieht folgendermassen aus:

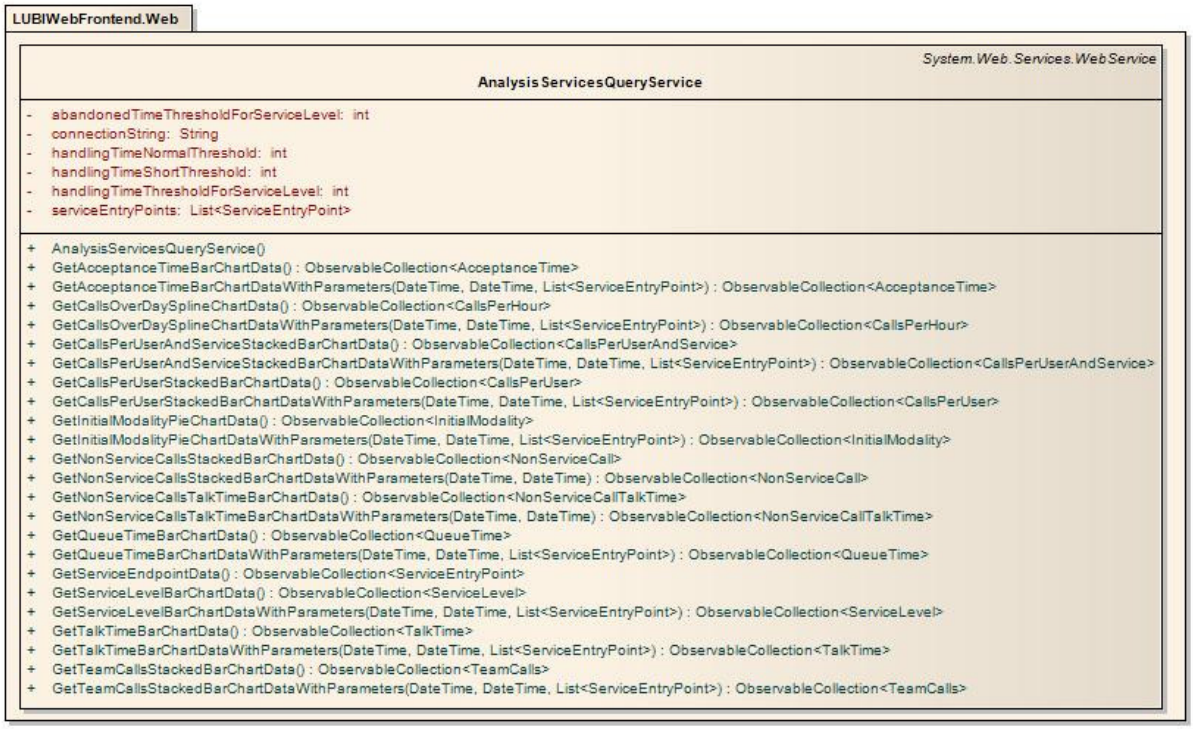


Abbildung 61: Klassendiagramm Namespace LUBIWebFrontend.Web

Durch die Annotierung der Klasse und Methoden generiert das Visual Studio automatisch die Beschreibung des Webservices in der Webservice Description Language (WSDL). Diese Beschreibung wird in der Common-Komponente für die Generierung der Webservice-Reference verwendet, welche durch den *AnalysisServicesQueryServiceWrapper* gekapselt wird.

Für den Transfer der Chartdaten werden im Namespace LUBIWebFrontend.Web.ProblemDomain für jedes Chart eine eigene Klasse erstellt. Sie enthält alle benötigten Daten für die Darstellung des Charts.

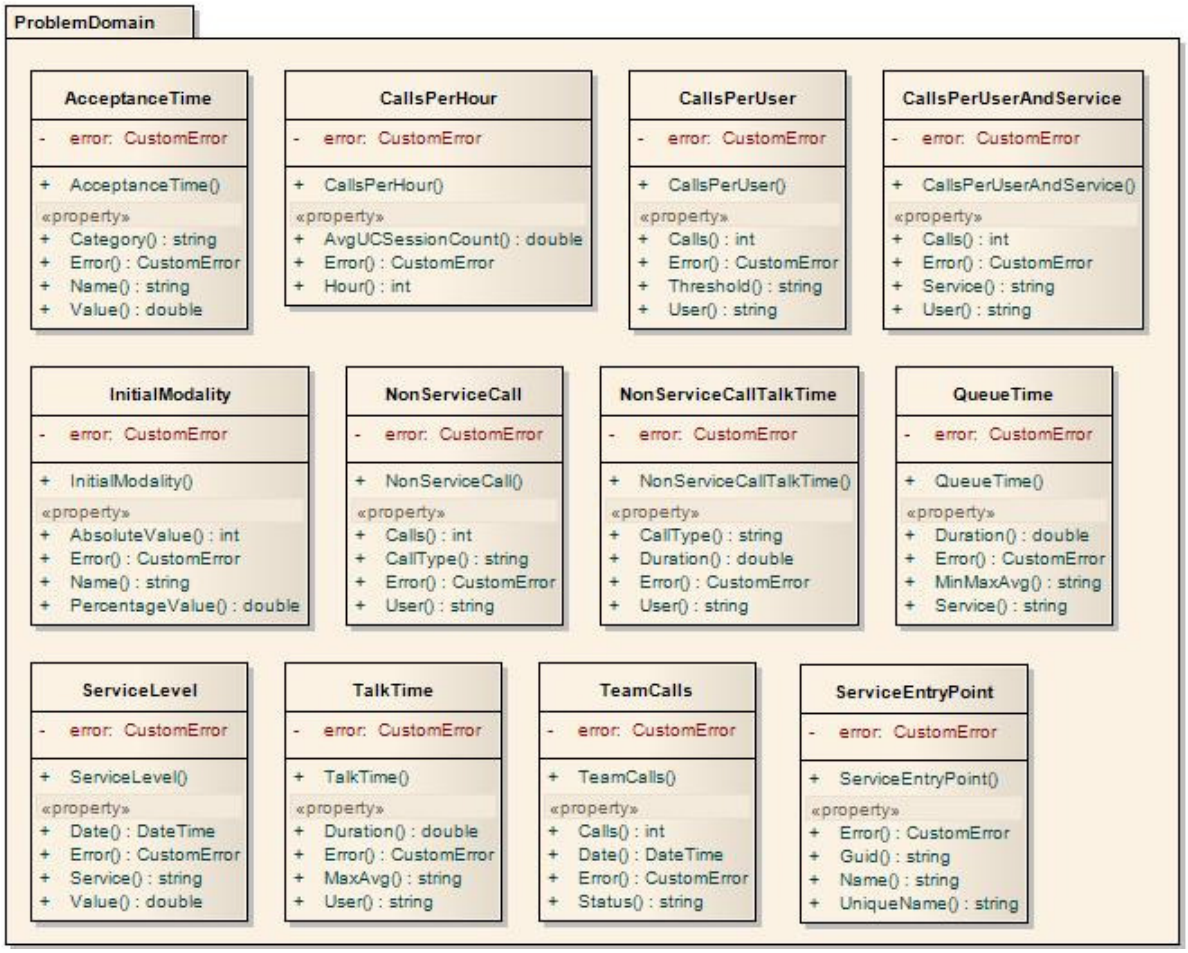


Abbildung 62: Klassendiagramm Namespace LUBIWebFrontend.Web.ProblemDomain

In der erstellten Applikation haben diese Klassen nur die Funktionalität von Data Transfer Objects, aber es ist denkbar, dass das Bedürfnis nach komplexeren Klassen in späteren Versionen aufkommt.

Für das ErrorHandling, sprich die Propagation von aufgetretenen Fehlern auf der WebService-Seite wurde eine eigene, zusätzliche Klasse erstellt, die relevante Informationen zum Fehler für den Transport zur Silverlight-Applikation kapselt.

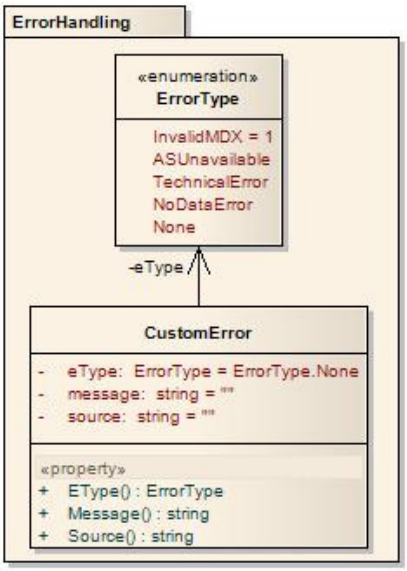


Abbildung 63: Klassendiagramm Namespace LUBIWebFronten.Web.ErrorHandling

9.5.4 LUBIWebFrontend.Module.DashboardModule

Die Komponente DashboardModule beinhaltet das UI und die dafür notwendigen Klassen für die Darstellung des Dashboards. Wie alle Module ist sie gleich aufgebaut: Im Namespace LUBIWebFrontend.Module.Dashboard liegt die Klasse *DashboardModule,* welches IModule, ein Interface aus der Prism-Bibliothek, implementiert. Diese Klasse initialisiert das Module und stösst die Anzeige der UI-Elemente an.

Im untergeordneten Namespace ViewModel findet sich die *DashboardViewModel*-Klasse welche gemäss MVVM-Pattern als Schnittstelle zwischen der View und dem Model dient. In diesem Fall ist das Model der AnalysisServicesQueryService. Im ViewModel finden sich sämtliche Daten, welche für die Anzeige des Dashboards benötigt werden. Diese Daten werden mittels Binding in die View eingebunden.

Im untergeordneten Namespace View sind die XAML-Deklaration des UI der Komponenten und ihre Code-Behind-Klasse abgelegt.

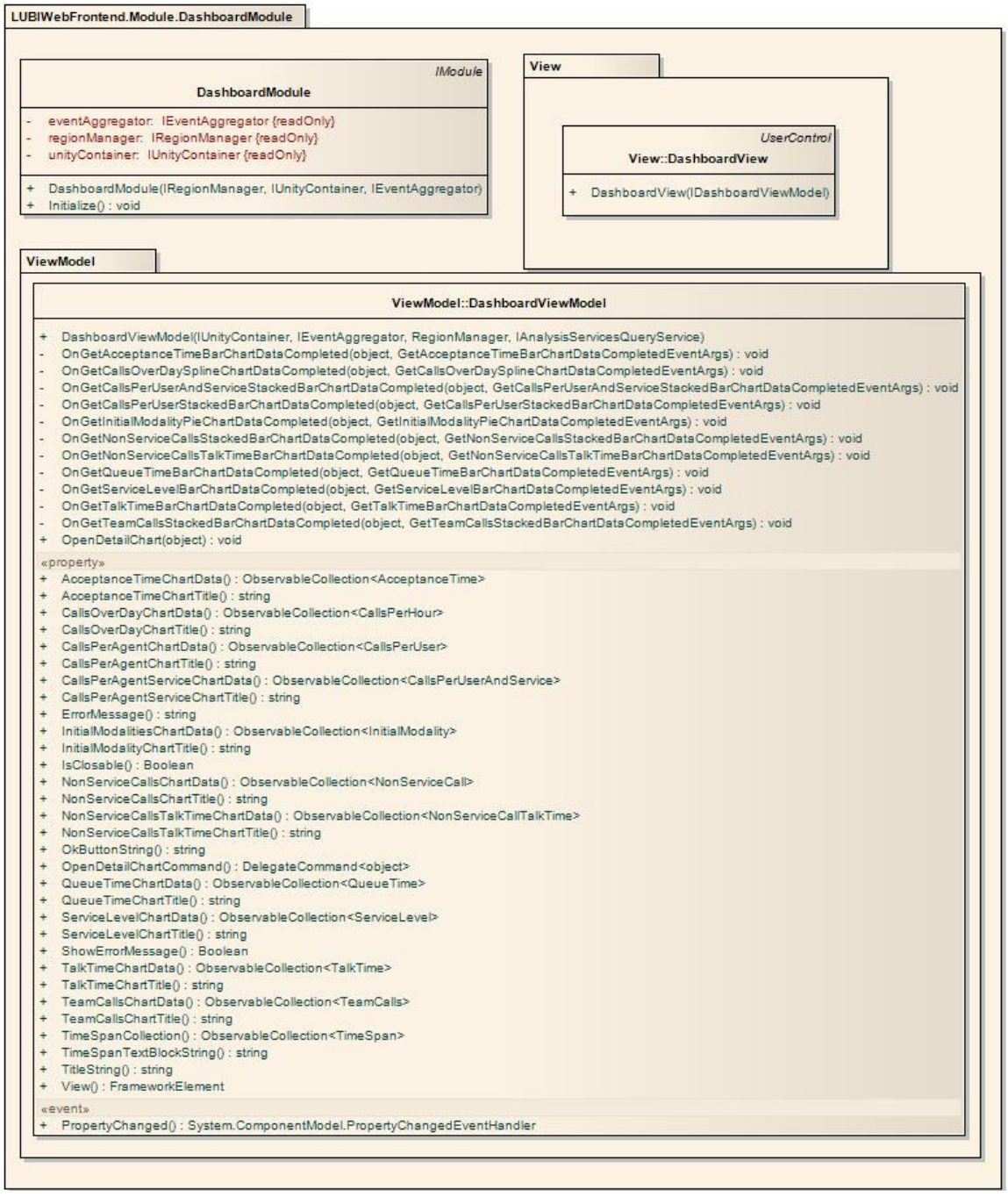


Abbildung 64: Klassendiagramm Namespace LUBIWebFrontend.Module.DashboardModule

9.5.5 LUBIWebFrontend.Module.InitialModalityChartModule

Diese Komponente wird stellvertretend für alle ChartModule im Folgenden beschrieben. Der Aufbau des InitialModality-Modules wie auch der restlichen ChartModule sind in der Struktur gleich wie das Dashboard-Module.

Die Klasse *InitialModalityChartModule,* welche das Interface IModule aus der Prism-Bibliothek implementiert, ist für die Initialisierung des Moduls zuständig. Im untergeordneten Namespace ViewModel liegt das *InitialModalityChartViewModel*. Im Namespace View liegt wiederum die XAML-Deklaration der View sowie die Code-Behind-Klasse der View.

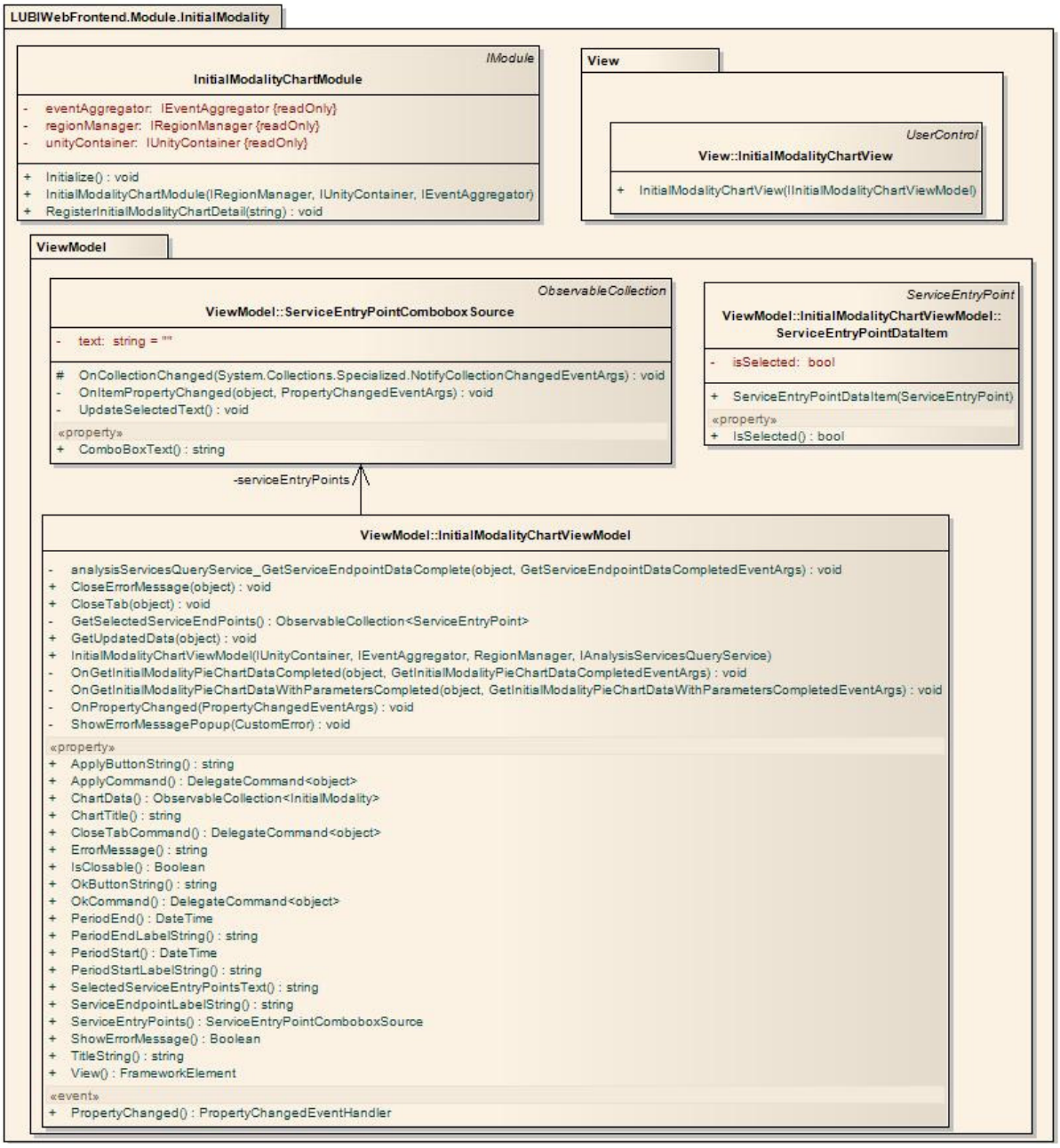


Abbildung 65: Klassendiagramm Namespace LUBIWebFrontend.Module.Dashbaord

Die innere Klasse *ServiceEntryPointComboboxSource*, welche von *ObservableCollection<ServiceEntryPointDataItem>* erbt, kapselt die Daten und Methoden, welche für die Anzeige einer speziell gestalteten Combobox verwendet werden.

9.6 Prozess und Thread-Modell

Silverlight arbeitet analog zu WPF mit einem UI-Thread. Wird dieser Thread zu stark belastet oder blockiert, kommt das Rendering ins Stocken. Deshalb werden langandauernde Operationen wie z.B. ein Service-Aufruf in einen Hintergrund-Thread ausgelagert und asynchron durchgeführt. Wenn das Resultat des Service-Calls zurückgeliefert wird, wird der Event im UI-

Thread ausgelöst und das Resultat in das UI eingebunden. Die Kommunkation mit dem Webservices des LUBIWebFrontend erfolgt deshalb asynchron.

10 Implementierung LUBI

10.1 Measure Berechnungen Service-UcSession

Wie bereits erwähnt, kann sich eine UcSession in verschiedenen Zuständen befinden. In der Faktentabelle *FactDim_ServiceUcSession* wird für jeden Zustand abgespeichert, wie lange sich die UcSession darin befunden hat. Die Berechnungen dieser Zeitdauern geschieht durch Berechnung der Differenz jeweils zweier Zeitstempel. Die folgende Abbildung zeigt, durch welche Zeitstempel die verschiedenen Zustände abgetrennt sind.

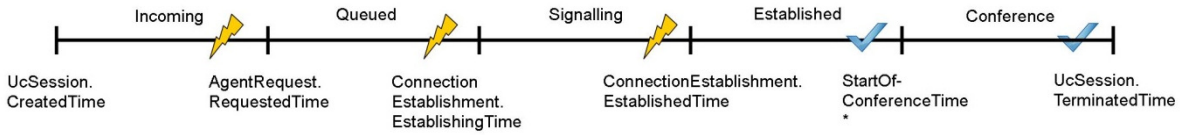


Abbildung 66: Zustände und Zeitstempel einer Service-UcSession

*Die StartOfConferenceTime entspricht dem Zeitstempel *CreatedTime* der dritten Conversation der UcSession.

Pro AgentRequest können mehrere ConnectionEstablishments erfolgen, wenn ein angefragter Kundenberater nicht reagiert, da er beispielsweise abwesend ist. Wie im Diagramm im Kapitel 8.4 zu sehen ist, können die Zustände Queued und Signalling mehrmals durchlaufen werden. Da in der Faktentabelle *FactDim_ServiceUcSession* jedoch nur ein Wert pro Zustand gespeichert werden kann, wird definiert, dass die Zeitstempel *EstablishingTime* und *EstablishedTime* des letzten, erfolgreichen ConnectionEstablishments für die Berechnung verwendet werden. Somit entspricht das Measure *TimeInQueuedState* der Zeit zwischen *AgentRequest.RequestedTime* und *ConnectionEstablishingTime* des letzten ConnectionEstablishment und das Measure *TimeInSignallingState* entspricht der *SignallingTime* des letzten ConnectionEstablishments.

Die folgende Tabelle zeigt auf, wie die verschiedenen Measures in der Faktentabelle *FactDim_ServiceUcSession* berechnet werden. Neben den Zeitdauern in den jeweiligen Zuständen werden auch noch weitere Measures berechnet, die für das Reporting relevant sind. Die Berechnungen erfolgen anhand der Zeitstempel in obiger Abbildung. Diese Zeitstempel sind in unterschiedlichen Tabellen der Live Datenbank abgelegt. Welche dies sind, ist in der Spalte *Berechnung* ersichtlich.

Measure	Berechnung
TimeInIncomingState	`AgentRequest.RequestedTime - UcSession.CreatedTime`
TimeInQueuedState	`ConnectionEstablishment.EstablishingTime – AgentRequest.RequestedTime`
TimeInSignallingState	`ConnectionEstablishment.EstablishedTime - ConnectionEstablishment.EstablishingTime`
TimeInEstablishedState	`StartOfConferenceTime - ConnectionEstablishment.EstablishedTime`
TimeInConferenceState	`UcSession.TerminatedTime - StartOfConferenceTime`
HandledInTime	`ConnectionEstablishment.EstablishedTime - AgentRequest.RequestedTime`
AbandondedInTime	`UcSession.TerminatedTime - UcSession.CreatedTime`
CallDuration	`TimeInIncomingState + TimeInQueuedState + TimeInSignallingState + TimeInEstablishedState + TimeInConferenceState`
TalkTime	`TimeInEstablishedState + TimeInConferenceState`
WaitTime	`TimeInQueuedState + TimeInSignallingState`

Tabelle 26: Measure Berechnungen Service-UcSession

10.2 Integration Services

Die Integration Services kommen beim Projekt LUBI zum Einsatz, um Daten aus der Live Datenbank auszulesen, für das DW zu transformieren und anzureichern und anschliessend in Form des Starschemas abzuspeichern. Hierbei wurden die für die Aufgabe erforderlichen Control Flows und Data Flows erstellt. Ebenfalls werden zwei Connection Manager benötigt.

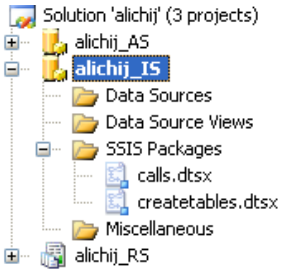


Abbildung 67: Solution alichij, Projekt alichij_IS

10.2.1 Connection Manager *SourceConnectionOLEDB*

Dieser ConnectionManager ist so eingerichtet, dass er das Lesen aus der Live Datenbank erlaubt. Wo immer innerhalb eines Datenflusses Daten aus der Live Datenbank gelesen werden müssen, kann auf diesen Connection Manager zurückgegriffen werden.

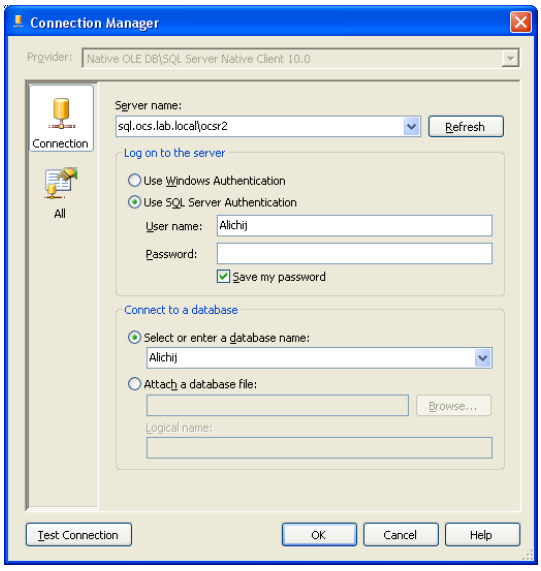


Abbildung 68: Connection Manager *SourceConnectionOLEDB*

10.2.2 ConnectionManager *DestinationConnectionOLEDB*

Dieser ConnectionManager ist so eingerichtet, dass er das Schreiben auf die Datenbank des DWs erlaubt. Wo immer innerhalb eines Datenflusses Daten in das DW gespeichert werden müssen, kann auf diesen Connection Manager zurückgegriffen werden.

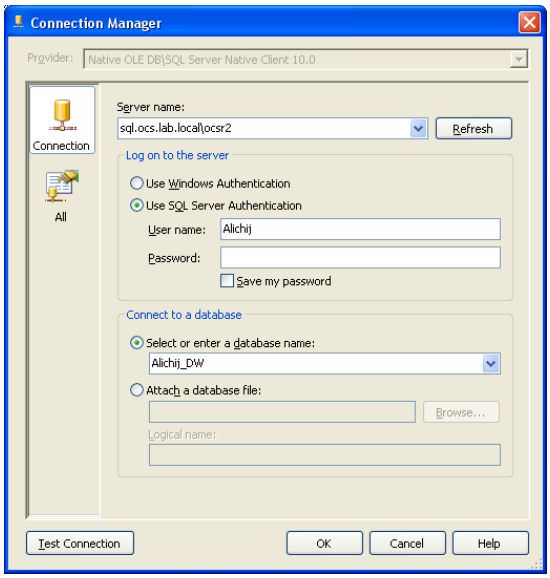


Abbildung 69: Connection Manager *DestinationConnectionOLEDB*

10.2.3 Erstellen der Tabellen im DW

Innerhalb des Integration Services Projekts *alichij_IS* wurde ein eigenes Package mit dem Namen *createtables.dtsx* erstellt, welches alle Control Flow Tasks enthält, die zum Erstellen der Tabellen im DW nötig sind. Theoretisch würde ein einziger SQL Task ausreichen, um alle Tabellen zu erstellen, der Übersichtlichkeit halber wurde aber für jede Tabelle ein eigener SQL Task erstellt. Wie in der folgenden Abbildung zu sehen ist, sind die Tasks nicht miteinander verbunden. Das bedeutet, dass die Tasks nicht in einer bestimmten Reihenfolge, sondern quasiparallel ablaufen. Die Tasks sind voneinander unabhängig, jeder erstellt eine Tabelle innerhalb der Datenbank des DWs.

Das Kopieren der Daten passiert hier noch nicht. Die meisten Tabellen bleiben nach dem Ausführen dieser SQL Tasks noch leer. Einige, wie zum Beispiel die Zeitdimensionstabellen werden mit Daten gefüllt, welche nicht aus der Datenbank geholt werden müssen, sondern ohne Datenbasis generiert werden können. Das Kopieren der Daten aus der Live Datenbank in das DW geschieht in einem anderen Package.

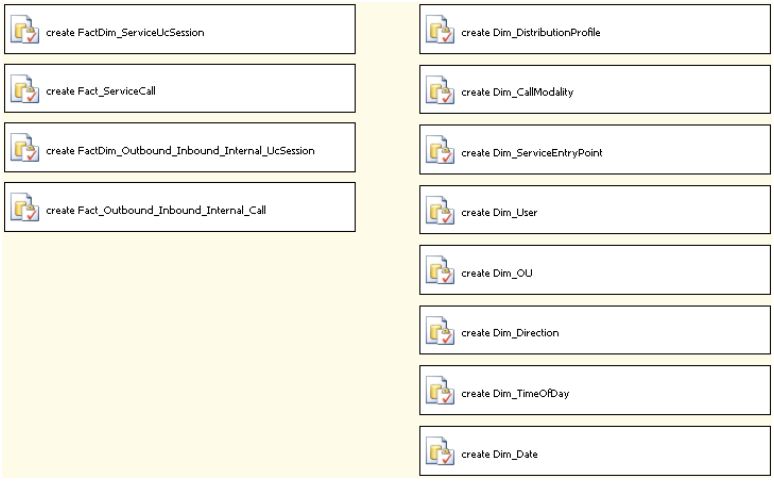


Abbildung 70: Control Flow Tasks

10.2.4 SQL Task create Dim_User

Die Tabelle *Dim_User* beinhaltet alle Kundenberater. Zusätzlich wird ein weiterer, nicht real existierender User benötigt, der anzeigt, dass einer UcSession eben kein Kundenberater zugeordnet wurde. Dies kann passieren, wenn die UcSession beendet wird, bevor ein Kundenberater gefunden wurde. Da es nicht erlaubt ist, dass sich in einer Faktentabelle leere Fremdschlüssel befinden, benötigt es einen solchen „Dummy-User“. Dieser User hat einen Primärschlüssel, der nur aus Nullen besteht und wird durch diesen SQL Task bereits der Tabelle hinzugefügt.

```
IF NOT EXISTS (SELECT Name FROM sys.tables WHERE name= 'Dim_User' AND type='U')
   BEGIN
     CREATE TABLE [dbo].[Dim_User] (
     [PK_UserId] uniqueidentifier NOT NULL,
     [Username] nvarchar(50) NOT NULL,
     [FirstName] nvarchar(50) NOT NULL,
     [LastName] nvarchar(50) NOT NULL,
     [FK_OrganizationUnitNodeId] uniqueidentifier NOT NULL,
     )

   INSERT INTO Alichij_DW.dbo.Dim_User
            ([PK_UserId]
            ,[Username]
            ,[FirstName]
            ,[LastName]
            ,[FK_OrganizationUnitNodeId]
            )
      VALUES
            ('00000000-0000-0000-0000-000000000000'
            ,'None'
            ,'None'
            ,'None'
            ,'00000000-0000-0000-0000-000000000000')
   END
```

10.2.5 SQL Task create Dim_Direction

Die Tabelle *Dim_Direction* wird nur für ausgehende, eingehende und interne UcSessions, aber nicht für Service-UcSessions verwendet und speichert ab, welcher dieser drei Typen die UcSession entspricht. Die Daten müssen nicht aus der Live Datenbank gelesen werden, da es sich nur um drei Datensätze handelt, die im Voraus bekannt sind. Sie werden durch den SQL Task gleich eingefügt.

```
IF EXISTS (SELECT name FROM sysobjects WHERE name = 'Dim_Direction' AND type = 'U')
   DROP TABLE [dbo].[Dim_Direction]

CREATE TABLE [dbo].[Dim_Direction] (
   [PK_Direction] uniqueidentifier NOT NULL,
   [Name] nvarchar(50) NOT NULL,
)

INSERT INTO [Alichij_DW].[dbo].[Dim_Direction]
           ([PK_Direction]
           ,[Name])
     VALUES
           ('00000000-0000-0000-0000-000000000000'
           ,'Internal')

INSERT INTO [Alichij_DW].[dbo].[Dim_Direction]
           ([PK_Direction]
           ,[Name])
     VALUES
           ('11111111-1111-1111-1111-111111111111'
           ,'Outbound')

INSERT INTO [Alichij_DW].[dbo].[Dim_Direction]
           ([PK_Direction]
           ,[Name])
     VALUES
           ('22222222-2222-2222-2222-222222222222'
           ,'Inbound')

INSERT INTO [Alichij_DW].[dbo].[Dim_Direction]
           ([PK_Direction]
           ,[Name])
     VALUES
           ('33333333-3333-3333-3333-333333333333'
           ,'Unknown')
```

10.2.6 SQL Task create Dim_TimeOfDay

Die Tabelle *Dim_TimeOfDay* beinhaltet alle Uhrzeiten eines Tages auf eine Sekunde genau. Dies bedeutet, dass die Tabelle 60*60*24=86400 Datensätze enthält, für jede Sekunde des Tages genau einen. Dieser SQL Task erstellt die Tabelle *Dim_TimeOfDay* und fügt anschliessend alle Uhrzeiten ein.

```
IF EXISTS (SELECT name FROM sysobjects WHERE name = 'Dim_TimeOfDay' AND type = 'U')
   DROP TABLE [dbo].[Dim_TimeOfDay]

CREATE TABLE [dbo].[Dim_TimeOfDay]
(
 TimeOfDay TIME PRIMARY KEY CLUSTERED,
 Hour TINYINT,
 Minute TINYINT,
 Second TINYINT,
)

DECLARE @CurrDate DATETIME
DECLARE @CurrTime TIME
SET @CurrDate = '01/01/1970'

WHILE @CurrDate < '01/02/1970'
 BEGIN
  SET @CurrTime = @CurrDate
  INSERT Dim_TimeOfDay
            (TimeOfDay
            , Hour
            , Minute
            , Second)
     VALUES
            (@CurrTime
            , DATENAME(hh, @CurrDate)
            , DATENAME(mi, @CurrDate)
            , DATENAME(ss, @CurrDate))
  SET @CurrDate = DATEADD(ss, 1, @CurrDate)
 END
```

10.2.6.1 SQL Task create Dim_Date

Die Dimensionstabelle *Dim_Date* ordnet jeder UcSession bzw. jedem Call ein eindeutiges Datum zu. Da Reportings jeweils für die letzten 5 Jahre erstellt werden, müssen alle Daten der letzten 5 Jahre in der Dimensionstabelle vorhanden sein. Die Tabelle umfasst also 365*5=1825 Datensätze. Dieser SQL Task erstellt die Dimensionstabelle *Dim_Date* und füllt die Tabelle mit den entsprechenden Daten.

```
IF EXISTS (SELECT name FROM sysobjects WHERE name = 'Dim_Date' AND type = 'U')
   DROP TABLE [dbo].[Dim_Date]

CREATE TABLE [dbo].[Dim_Date]
(
   Date DATE PRIMARY KEY CLUSTERED,
   Day VARCHAR(10),
   Month VARCHAR(10),
   Year SMALLINT,
   DayOfMonth TINYINT,
   DayOfYear SMALLINT,
   WeekOfYear TINYINT,
   MonthOfYear TINYINT,
   QuarterOfYear VARCHAR(2),
)

DECLARE @CurrDate DATETIME
DECLARE @EndDate DATETIME

SET @CurrDate = DATEADD(yy, -5, GETDATE())
SET @EndDate = GETDATE()

WHILE @CurrDate < @EndDate
 BEGIN
  INSERT Dim_Date
            (Date
            , Day
            , Month
            , Year
            , DayOfMonth
            , DayOfYear
            , WeekOfYear
            , MonthOfYear
            , QuarterOfYear)
  VALUES
            (@CurrDate
            , DATENAME(dw, @CurrDate)
            , DATENAME(month, @CurrDate)
            , YEAR(@CurrDate)
            , DAY(@CurrDate)
            , DATENAME(DY, @CurrDate)
            , DATENAME(wk, @CurrDate)
```

```
            , MONTH(@CurrDate)
            , 'Q' + DATENAME(quarter,@CurrDate))
 SET @CurrDate = DATEADD(dd, 1, @CurrDate)
END
```

	Date	Day	Month	Year	DayOfMonth	DayOfYear	WeekOfYear	MonthOfYear	QuarterOfYear
1	2005-04-23	Saturday	April	2005	23	113	17	4	Q2
2	2005-04-24	Sunday	April	2005	24	114	18	4	Q2
3	2005-04-25	Monday	April	2005	25	115	18	4	Q2
4	2005-04-26	Tuesday	April	2005	26	116	18	4	Q2
5	2005-04-27	Wednesday	April	2005	27	117	18	4	Q2
6	2005-04-28	Thursday	April	2005	28	118	18	4	Q2
7	2005-04-29	Friday	April	2005	29	119	18	4	Q2
8	2005-04-30	Saturday	April	2005	30	120	18	4	Q2
9	2005-05-01	Sunday	May	2005	1	121	19	5	Q2
10	2005-05-02	Monday	May	2005	2	122	19	5	Q2

Abbildung 71: Dimensionstabelle Dim_Date (Ausschnitt)

Auf die Auflistung der SQL Statements der restlichen SQL Tasks wird hier verzichtet, da bei allen anderen Tabellen keine Daten eingefügt, sondern nur die jeweiligen Tabellen erstellt werden. Wie diese Tabellen im Detail aussehen, wird in den Kapiteln 8.4.4 bzw. 8.5.4 beschrieben.

10.2.7 ETL Vorgang

Ein weiteres Package *calls.dtsx* wurde erstellt, um den gesamten ETL-Vorgang und verwandte Arbeiten zu realisieren. Das Package enthält einen Control Flow mit drei Tasks. Der erste Task *delete old data* löscht alle UcSessions, die älter als 5 Jahre sind und alle dazugehörigen Calls aus dem DW. Beim zweiten Task *copy new data* handelt es sich um einen Control Flow Task, der den eigentlichen ETL-Vorgang darstellt. Er wird als Container für eine grosse Zahl von Control Flow Task Komponenten verwendet und ist sehr umfangreich. Der dritte und letzte Task schliesslich führt das Cube Processing aus. Der Cube holt sich somit die neu integrierten Daten aus dem DW.

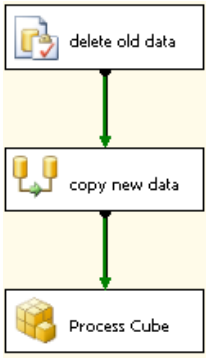


Abbildung 72: Control Flow Tasks von *calls.dtsx*

Der Data Flow Task *copy new data* ist wie erwähnt sehr umfangreich und soll noch detaillierter erklärt werden.

Im Data Flow Task werden die benötigten Daten aus der Live Datenbank kopiert, in die richtige Struktur gebracht und danach im DW abgespeichert. Dazu wird als erste Komponente im Data Flow Task eine OLE DB Source definiert, mit welcher definierte Spalten aus der Live Datenbank genommen und dem Data Flow hinzugefügt werden. Mittels mehreren Lookup-Komponenten werden dann Joins auf der Live Datenbank gemacht, um dem Data Flow weitere Spalten hinzuzufügen. Weiter werden dann Derived Column Komponenten verwendet, um weitere Spalten zu berechnen. Am Ende des Data Flow Tasks werden die Daten mit einer OLE DB Destination Komponente in das DW geschrieben.

Da mehrere Tabellen der Live Datenbank relevant sind und die Daten auch in unterschiedliche Tabellen des DWs geschrieben werden, sind mehrere Sources bzw. Destinations nötig, was auch zu mehreren Data Flows führt.

Aufgrund der grossen Anzahl von Komponenten im Control Flow Task würde die detaillierte Beschreibung jeder einzelnen Komponente den Rahmen dieses Dokuments sprengen. Für eine Übersicht aller Komponenten der Integration Services verweisen wir deshalb auf den Anhang. Um eine einfachere Orientierung zu ermöglichen, wurde allerdings eine Konvention bei der Benennung der Komponenten eingehalten. Diese Benennung ist abhängig von der Art der Komponente:

Komponente	Benennung	Beispiel
OLE DB Source	OLE DB Source <Tabellenname>	OLE DB Source UcSession
OLE DB Destination	OLE DB Destination <Tabellenname>	OLE DB Destination FactDim_ServiceUcSession
ConditionalSplit	Is<Condition>	IsTerminated
Lookup	<NameOfNewColumn>, <NameOfAnotherNewColumn>	EstablishedTime, EstablishingTime
DerivedColumn	<NameOfNewColumn>, <NameOfAnotherNewColumn>	TimeInIncomingState, TimeInSignallingState, TimeInConferenceState
Multicast	Keine besondere Konvention, möglichst gute Benennung des bisherigen DataFlows	ServiceUcSessions
Sort	Keine besondere Konvention, möglichst gute Benennung des bisherigen DataFlows	SortedServiceUcSessions

Tabelle 27: Konvention Komponentenbennenung ETL

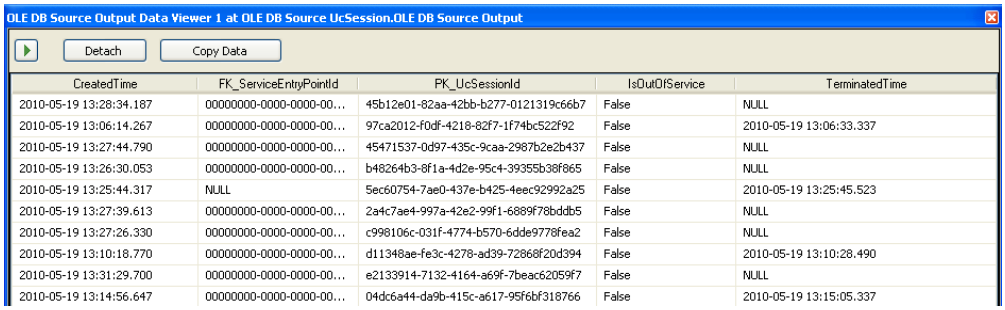
Eine allgemeine Beschreibung der Komponenten befindet sich in Kapitel 6.1.1.3

Im Folgenden wird das Prinzip anhand eines Beispiels beschrieben. Alle Komponenten, die benötigt werden, um die UcSessions aus der Live Datenbank zu lesen, zu transformieren und in das DW abzuspeichern, werden detaillierter beschrieben. Bei der Übersicht im Anhang handelt es sich um den rot umrahmten Teil. Auf eine Beschreibung der restlichen Komponenten verzichten wir grösstenteils, um den Rahmen nicht zu sprengen und um uns nicht zu wiederholen.

10.2.7.1 Komponente *OLE DB Source UcSession* (OLE DB Source)

Diese Komponente verwendet den Connection Manager *SourceConnectionOLEDB* und liest aus der Tabelle *UcSession* der Live Datenbank folgende Spalten aller Datensätze und fügt sie dem Datenfluss hinzu:

- CreatedTime
- PK_UcSessionId
- TerminateTime
- FK_ServiceEntryPointId
- IsOutOfService

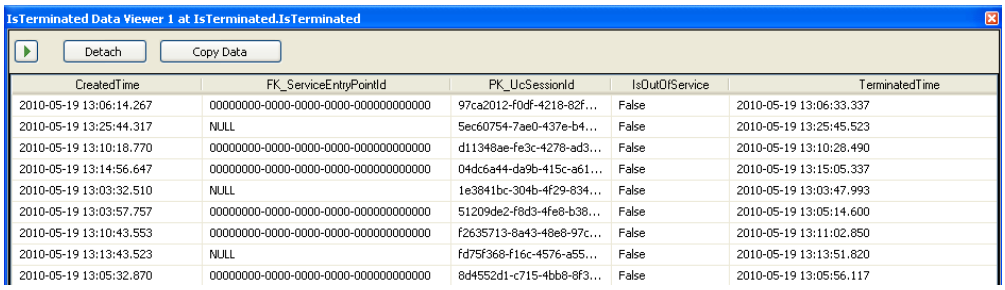


CreatedTime	FK_ServiceEntryPointId	PK_UcSessionId	IsOutOfService	TerminatedTime
2010-05-19 13:28:34.187	00000000-0000-0000-00…	45b12e01-82aa-42bb-b277-0121319c66b7	False	NULL
2010-05-19 13:06:14.267	00000000-0000-0000-00…	97ca2012-f0df-4218-82f7-1f74bc522f92	False	2010-05-19 13:06:33.337
2010-05-19 13:27:44.790	00000000-0000-0000-00…	45471537-0d97-435c-9caa-2987b2e2b437	False	NULL
2010-05-19 13:26:30.053	00000000-0000-0000-00…	b48264b3-8f1a-4d2e-95c4-39355b38f865	False	NULL
2010-05-19 13:25:44.317	NULL	5ec60754-7ae0-437e-b425-4eec92992a25	False	2010-05-19 13:25:45.523
2010-05-19 13:27:39.613	00000000-0000-0000-00…	2a4c7ae4-997a-42e2-99f1-6889f78bddb5	False	NULL
2010-05-19 13:27:26.330	00000000-0000-0000-00…	c998106c-031f-4774-b570-6dde9778fea2	False	NULL
2010-05-19 13:10:18.770	00000000-0000-0000-00…	d11348ae-fe3c-4278-ad39-72868f20d394	False	2010-05-19 13:10:28.490
2010-05-19 13:31:29.700	00000000-0000-0000-00…	e2133914-7132-4164-a69f-7beac62059f7	False	NULL
2010-05-19 13:14:56.647	00000000-0000-0000-00…	04dc6a44-da9b-415c-a617-95f6bf318766	False	2010-05-19 13:15:05.337

Abbildung 73: DataViewer nach Komponente *OLE DB Source UcSession*

10.2.7.2 Komponente *IsTerminated* (Conditional Split)

Um Inkonsistenzen zu vermeiden, werden nur UcSessions in das DW kopiert, die bereits terminiert sind. Diese Komponente sortiert alle UcSessions aus, die noch nicht terminiert wurden, in der Spalte *TerminatedTime* also nicht NULL sind.



CreatedTime	FK_ServiceEntryPointId	PK_UcSessionId	IsOutOfService	TerminatedTime
2010-05-19 13:06:14.267	00000000-0000-0000-0000-000000000000	97ca2012-f0df-4218-82f...	False	2010-05-19 13:06:33.337
2010-05-19 13:25:44.317	NULL	5ec60754-7ae0-437e-b4...	False	2010-05-19 13:25:45.523
2010-05-19 13:10:18.770	00000000-0000-0000-0000-000000000000	d11348ae-fe3c-4278-ad3...	False	2010-05-19 13:10:28.490
2010-05-19 13:14:56.647	00000000-0000-0000-0000-000000000000	04dc6a44-da9b-415c-a61...	False	2010-05-19 13:15:05.337
2010-05-19 13:03:32.510	NULL	1e3841bc-304b-4f29-834...	False	2010-05-19 13:03:47.993
2010-05-19 13:03:57.757	00000000-0000-0000-0000-000000000000	51209de2-f8d3-4fe8-b38...	False	2010-05-19 13:05:14.600
2010-05-19 13:10:43.553	00000000-0000-0000-0000-000000000000	f2635713-8a43-48e8-97c...	False	2010-05-19 13:11:02.850
2010-05-19 13:13:43.523	NULL	fd75f368-f16c-4576-a55...	False	2010-05-19 13:13:51.820
2010-05-19 13:05:32.870	00000000-0000-0000-0000-000000000000	8d4552d1-c715-4bb8-8f3...	False	2010-05-19 13:05:56.117

Abbildung 74: DataViewer nach Komponente *IsTerminated*

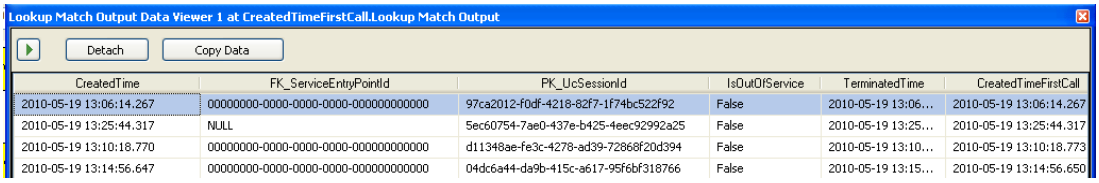
10.2.7.3 Komponente *CreatedTimeFirstCall* (Lookup)

Neue Spalten	CreatedTimeFirstCall
Übereinstimmungsfeld	(DF) PK_UcSessionId == (DB) FK_UcSessionId[28]

Diese Komponente fügt die CreatedTime des ersten Calls der UcSession dem Datenfluss hinzu. Dies geschieht mit folgendem SQL Statement, welches auf der Live Datenbank abgesetzt wird:

```
SELECT dbo.Conversation.FK_UcSessionId, MIN(dbo.Call.CreatedTime) AS CreatedTimeFirstCall
FROM dbo.Call INNER JOIN dbo.Conversation
ON dbo.Call.FK_ConversationId = dbo.Conversation.PK_ConversationId
GROUP BY dbo.Conversation.FK_UcSessionId
```

Das Ergebnis des SQL Statements wird mit dem bisherigen Datenfluss über das Feld *UcSessionId* verknüpft, wie obige Tabelle zeigt.



CreatedTime	FK_ServiceEntryPointId	PK_UcSessionId	IsOutOfService	TerminatedTime	CreatedTimeFirstCall
2010-05-19 13:06:14.267	00000000-0000-0000-0000-000000000000	97ca2012-f0df-4218-82f7-1f74bc522f92	False	2010-05-19 13:06...	2010-05-19 13:06:14.267
2010-05-19 13:25:44.317	NULL	5ec60754-7ae0-437e-b425-4eec92992a25	False	2010-05-19 13:25...	2010-05-19 13:25:44.317
2010-05-19 13:10:18.770	00000000-0000-0000-0000-000000000000	d11348ae-fe3c-4278-ad39-72868f20d394	False	2010-05-19 13:10...	2010-05-19 13:10:18.773
2010-05-19 13:14:56.647	00000000-0000-0000-0000-000000000000	04dc6a44-da9b-415c-a617-95f6bf318766	False	2010-05-19 13:15...	2010-05-19 13:14:56.650

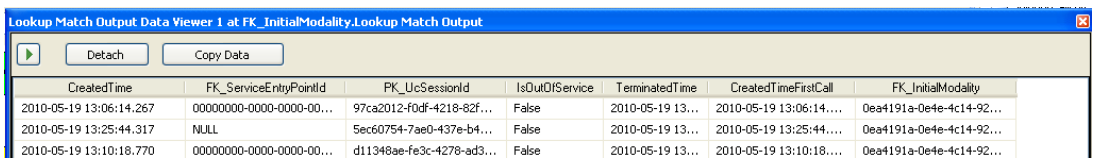
Abbildung 75: DataViewer nach Komponente *CreatedTimeFirstCall*

10.2.7.4 Komponente *FK_InitialModality* (Lookup)

Neue Spalten	FK_InitialModality
Übereinstimmungsfelder	(DF) PK_UcSessionId == (DB) FK_UcSessionId (DF) CreatedTimeFirstCall == (DB) CreatedTime

Diese Komponente fügt die Initial Modality der UcSession dem Datenfluss hinzu. Dazu wird der folgende SQL-Befehl auf der Live Datenbank abgesetzt und wie in der obigen Tabelle zu sehen mit dem Datenfluss verknüpft:

```
SELECT dbo.Call.CreatedTime, dbo.Conversation.FK_UcSessionId, dbo.Call.FK_CallModalityId
FROM dbo.Call INNER JOIN dbo.Conversation
ON dbo.Call.FK_ConversationId = dbo.Conversation.PK_ConversationId
```



CreatedTime	FK_ServiceEntryPointId	PK_UcSessionId	IsOutOfService	TerminatedTime	CreatedTimeFirstCall	FK_InitialModality
2010-05-19 13:06:14.267	00000000-0000-0000-00...	97ca2012-f0df-4218-82f...	False	2010-05-19 13...	2010-05-19 13:06:14....	0ea4191a-0e4e-4c14-92...
2010-05-19 13:25:44.317	NULL	5ec60754-7ae0-437e-b4...	False	2010-05-19 13...	2010-05-19 13:25:44....	0ea4191a-0e4e-4c14-92...
2010-05-19 13:10:18.770	00000000-0000-0000-00...	d11348ae-fe3c-4278-ad3...	False	2010-05-19 13...	2010-05-19 13:10:18....	0ea4191a-0e4e-4c14-92...

Abbildung 76: DataViewer nach Komponente *FK_InitialModality*

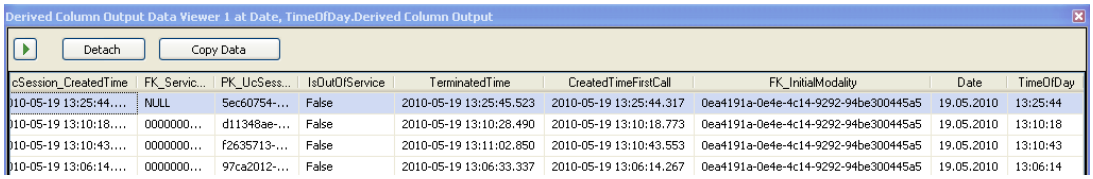
[28] Die Felder, auf denen der Join vergleicht, stammen entweder aus dem Datenfluss (DF), der produktiven Datenbank (DB) oder dem Data Warehouse (DW). Die Felder aus der produktiven Datenbank werden meist durch ein SQL Statement erhalten.

10.2.7.5 Komponente *Date, TimeOfDay* (Derived Column)

Die Spalten *Date* und *TimeOfDay* werden aus der Spalte *CreatedTime* berechnet und dem Datenfluss hinzugefügt. Die Spalte *CreatedTime*, welche den Datentyp datetime hat, wird in die Typen DT_DBDATE bzw. DT_DBTIME2 gecastet, so dass jeweils die Zeit bzw. das Datum abgeschnitten wird und die gewünschte Information übrig bleibt.

Date:
```
(DT_DBDATE)CreatedTime
```

TimeOfDay:
```
(DT_DBTIME2,0)CreatedTime
```



cSession_CreatedTime	FK_Servic...	PK_UcSess...	IsOutOfService	TerminatedTime	CreatedTimeFirstCall	FK_InitialModality	Date	TimeOfDay
010-05-19 13:25:44....	NULL	5ec60754-...	False	2010-05-19 13:25:45.523	2010-05-19 13:25:44.317	0ea4191a-0e4e-4c14-9292-94be300445a5	19.05.2010	13:25:44
010-05-19 13:10:18....	0000000...	d11348ae-...	False	2010-05-19 13:10:28.490	2010-05-19 13:10:18.773	0ea4191a-0e4e-4c14-9292-94be300445a5	19.05.2010	13:10:18
010-05-19 13:10:43....	0000000...	f2635713-...	False	2010-05-19 13:11:02.850	2010-05-19 13:10:43.553	0ea4191a-0e4e-4c14-9292-94be300445a5	19.05.2010	13:10:43
010-05-19 13:06:14....	0000000...	97ca2012-...	False	2010-05-19 13:06:33.337	2010-05-19 13:06:14.267	0ea4191a-0e4e-4c14-9292-94be300445a5	19.05.2010	13:06:14

Abbildung 77: DataViewer nach Komponente *Date, TimeOfDay*

10.2.7.6 Komponente *IsServiceUcSession* (ConditionalSplit)

Diese Komponente teilt den Datenfluss auf. Die Komponente hat zwei Ausgänge. Alle Service-UcSessions werden auf den ersten Ausgang und alle weiteren UcSessions auf den zweiten Ausgang geleitet. Service-UcSessions werden als Service-UcSessions erkannt, wenn *FK_ServiceEntryPoint* nicht NULL ist:

```
!ISNULL(FK_ServiceEntryPointId)
```

10.2.7.7 Komponente *EstablishedTime, EstablishingTime* (Lookup)

Neue Spalten	EstablishedTime, EstablishingTime
Übereinstimmungsfeld	(DF) PK_UcSessionId == (DB) FK_UcSessionId

```
SELECT AR.FK_UcSessionId, AR.RequestedTime, CE.EstablishedTime, CE.EstablishingTime, CE.Count,
   AR.FK_DistributionProfileAtSolvedTimeId
FROM dbo.AgentRequest AS AR INNER JOIN
   (SELECT FK_AgentRequestId, MAX(EstablishedTime) AS EstablishedTime, MAX(EstablishingTime) AS
      EstablishingTime, COUNT(FK_AgentRequestId) AS Count
   FROM dbo.ConnectionEstablishment
   GROUP BY FK_AgentRequestId) AS CE
ON AR.PK_AgentRequestId = CE.FK_AgentRequestId
```

10.2.7.8 Komponente *FK_UserId* (Lookup)

Neue Spalten	FK_UserId
Übereinstimmungsfeld	(DF) PK_UcSessionId == (DB) FK_UcSessionId

```
SELECT dbo.AgentRequest.FK_UcSessionId, dbo.ConnectionEstablishment.FK_UserId
FROM dbo.AgentRequest INNER JOIN dbo.ConnectionEstablishment
ON dbo.AgentRequest.PK_AgentRequestId = dbo.ConnectionEstablishment.FK_AgentRequestId
```

10.2.7.9 Komponente *UserIdNullCheck* (Derived Column)

Die Komponente überprüft, ob die UserId Null ist. Da die UserId nicht Null sein darf, da in der Faktentabelle keine leeren Fremdschlüssel erlaubt sind, wird bei leerer UserId die UserId auf den Standardwert 00000000-0000-0000-0000-000000000000 gesetzt. Dieser Wert ist in der Variablen *NoUser* abgelegt.
```
ISNULL(FK_UserId) ? (DT_GUID)@[User::NoUser] : FK_UserId
```

10.2.7.10 Komponente *ServiceUcSessions* (Multicast)

Diese Komponente dupliziert den Datenfluss.

10.2.7.11 Komponente *ConferenceStartTime* (Lookup)

Neue Spalten	ConferenceStartTime
Übereinstimmungsfeld	(DF) PK_UcSessionId == (DB) FK_UcSessionId

Die ConferenceStartTime entspricht der CreatedTime der als drittes erstellten Conversation der UcSession.

```
SELECT *
FROM
   (SELECT FK_UcSessionId, PK_ConversationId, CreatedTime,
      Rank() OVER (Partition BY FK_UcSessionId ORDER BY CreatedTime) AS Rank
FROM dbo.Conversation) tmp
WHERE Rank = 3
```

10.2.7.12 Komponente *New ServiceUcSessions* (Lookup)

Neue Spalten	keine
Übereinstimmungsfeld	(DF) PK_UcSessionId == (DW) FactDim_ServiceUcSession.UcSessionId

Die Komponente überprüft, ob ein Datensatz des Datenflusses bereits im DW vorhanden ist. Um zu verhindern, dass UcSessions bei mehrfacher Ausführung des Packages mehrfach in das DW übernommen werden, werden die UcSessions aus dem Datenfluss entfernt, die bereits im DW sind. Dies geschieht mittels einem Lookup. Als linke Spalte wird das Feld *PK_UcSessionId* des Datenflusses genommen, als rechte Spalte das Feld *UcSessionId* der Tabelle *FactDim_ServiceUcSession* im DW. Im Fall, dass ein Datensatz bereits im DW vorhanden ist, resultiert ein Treffer. Es müssen also alle Datensätze an die nächste Komponente weitergegeben werden, bei denen kein Treffer resultiert. Für die weitere Verwendung wird also der No Match Output der Komponente verwendet.

10.2.7.13 Komponente *TimeInIncomingState, TimeInSignallingState, TimeInConferenceState* (Derived Column)

Bei der Berechnung der Zeitdauern, welche die UcSession in den unterschiedlichen Zuständen verbracht hat, werden die Zeitstempel verwendet, die im Abschnitt 10.1 aufgezeigt wurden. Grundsätzlich entsprechen die Zeitdauern der Differenz zweier Zeitstempel. Da diese Zeitstempel NULL sein können, wenn bestimmte Zustände gar nicht erreicht werden, gibt es bei der Berechnung drei Fälle zu unterscheiden.

Für die folgende Tabelle gilt:

- *StartTimeStamp* speichert die Zeit, in der in einen Zustand gewechselt wird.
- *EndTimeStamp* speichert die Zeit, in der ein Zustand verlassen wird.
- Es wird die Zeit berechnet, die in dem Zustand verbracht wurde.

Fall	**Berechnung**
StartTimeStamp und *EndTimeStamp* sind nicht NULL	Die Zeitdauer wird berechnet durch *EndTimeStamp* – *StartTimeStamp*
Nur der *EndTimeStamp* ist NULL	*EndTimeStamp* kann nicht für die Berechnung verwendet werden. Die Zeit wird berechnet durch *UcSession.TerminatedTime* – *StartTimeStamp*. Dies ist korrekt, da der nächste Zustand nicht erreicht wurde und daher die *TerminatedTime* der UcSession mit dem Ende des Zustands gleichgesetzt werden kann.

StartTimeStamp und *EndTimeStamp* sind NULL	Der Zustand wurde nie erreicht und daher ist die darin verbrachte Zeit 0.

Tabelle 28: Fallunterscheidungen bei Zeitdauerberechnungen

Die folgenden vier Felder werden durch diese Komponente dem Datenfluss hinzugefügt. Die drei Fallunterscheidungen werden berücksichtigt.

TimeInIncomingState:
```
ISNULL(AR_RequestedTime) ? ISNULL(UcSession_CreatedTime) ? 0 :
DATEDIFF("ms",UcSession_CreatedTime,AR_CancelledTime) :
DATEDIFF("ms",UcSession_CreatedTime,AR_RequestedTime)
```

TimeInQueuedState:
```
(ISNULL(CE_EstablishingTime) ? ISNULL(AR_RequestedTime) ? 0 :
DATEDIFF("ms", AR_RequestedTime,TerminatedTime) :
DATEDIFF("ms", AR_RequestedTime,CE_EstablishingTime))
```

TimeInSignallingState
```
ISNULL(AR_SolvedTime) ? ISNULL(AR_RequestedTime) ? 0 :
DATEDIFF("ms",AR_RequestedTime,AR_CancelledTime) - TimeInSignallingState :
DATEDIFF("ms",AR_RequestedTime,AR_SolvedTime) - TimeInSignallingState
```

TimeInConferenceState:
```
(ISNULL(ConferenceStartTime) ? 0 : DATEDIFF("ms", ConferenceStartTime, TerminatedTime))
```

10.2.7.14 Komponente *TimeInEstablishedState* (Derived Column)

Die Zeitdauer im EstablishedState wird von dieser Komponente berechnet. Da die TimeInConferenceState für die Berechnung benötigt wird, konnte diese Berechnung nicht bereits in der vorherigen Komponente durchgeführt werden.

TimeInEstablishedState:
```
(ISNULL(CE_EstablishedTime) ? 0 :
DATEDIFF("ms", CE_EstablishedTime,TerminatedTime) – TimeInConferenceState)
```

10.2.7.15 Komponente *HandledInTime, AbandonedInTime, CallDuration, Talk Time, WaitTime* (Derived Column)

Neben den Zeitdauern in den verschiedenen Zuständen werden auch noch weitere Zeitdauern berechnet, die für das Reporting wichtig sind:

HandledInTime:
```
TimeInQueuedState + TimeInSignallingState
```

AbandonedInTime:
```
DATEDIFF("ms",UcSession_CreatedTime,TerminatedTime)
```

UcSessionDuration:
```
TimeInIncomingState + TimeInQueuedState + TimeInSignallingState + TimeInEstablishedState +
TimeInConferenceState
```

TalkTime:
```
TimeInEstablishedState + TimeInConferenceState
```

WaitTime:
```
TimeInQueuedState + TimeInSignallingState
```

10.2.7.16 Komponente *OLE DB Destination FactDim_ServiceUcSession*

Die Komponente verwendet den ConnectionManager *DestinationConnectionOLEDB* und schreibt die Daten aus dem Datenfluss in die Tabelle *FactDim_ServiceUcSession* des DWs. Die Zuordnung der Spalten des Datenflusses zu den Spalten in der Tabelle *FactDim_ServiceUcSession* sieht folgendermassen aus:

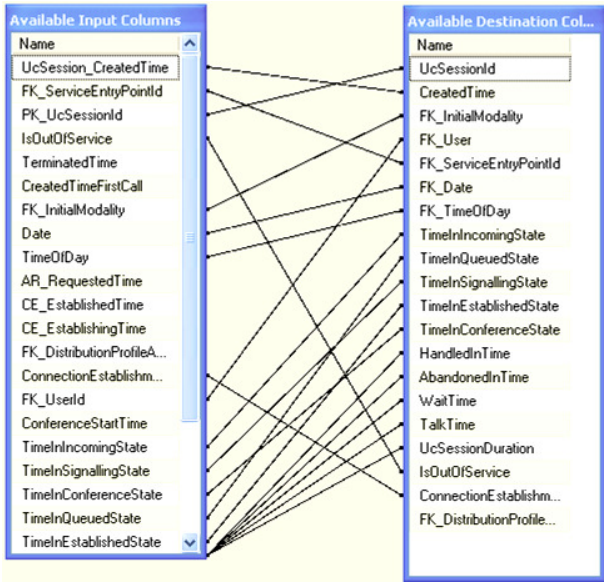


Abbildung 78: OLE DB Destination, Seite *Mappings*

Damit ist die Beschreibung aller Komponenten des Bereichs *ServiceUcSessions* abgeschlossen (roter Bereich in der Komponentenübersicht im Anhang)

10.2.8 Variablen

Im SSIS Paket *calls.dtsx* wurden einige Variablen gespeichert, auf die in einigen Data Flow Task Komponenten lesend zugegriffen wird:

Name	Data Flow Task	Datentyp	Wert	Verwendung in Komponente(n)
Outbound	copy new data	String	{11111111-1111-1111-1111-11111111111}	Direction
Inbound	copy new data	String	{22222222-2222-2222-2222-222222222222}	Direction
Internal	copy new data	String	{00000000-0000-0000-0000-000000000000}	Direction
UnknownDirection	copy new data	String	{33333333-3333-3333-3333-333333333333}	Direction
NoUser	copy new data	String	{00000000-0000-0000-0000-000000000000}	UserIdNullCheck, FK_UserIdsNullCheck
NoOU	copy new data	String	{00000000-0000-0000-0000-000000000000}	OUNodeNullCheck

Tabelle 29: Übersicht Variablen

Die Variablen *Outbound*, *Inbound*, *Internal* und *UnknownDirection* speichern Primärschlüssel (unique identifier) der Dimensionstabelle *Direction* ab. In der Komponente *Direction* des Data Flow Tasks wird jeder ausgehenden, eingehenden oder internen UcSession einer dieser vier Werte zugewiesen. Die Entscheidung, welcher dieser vier Werte zugewiesen wird, wird durch den SipCaller und SipCallee der Conversation gefällt:

Bedingung	Wert
SipCaller und SipCallee entsprechen Usern (Kundenberater)	Internal
SipCaller entspricht User (Kundenberater)	Outbound
SipCallee entspricht User (Kundenberater)	Inbound
sonst	UnknownDirection

Die Variable *NoUser* hält die ID des “Dummy-Users” (siehe Kapitel 10.2.4). Diese wird in der Komponente *UserIdNullCheck* für alle Service-UcSessions bzw. in der Komponente *FK_UserIdsNullCheck* für alle ausgehenden, eingehenden oder internen UcSessions gesetzt, wenn bisher keine andere UserId gesetzt ist. Dies wird gemacht, um leere Fremdschlüssel in den Faktentabellen zu verhindern.

Die Variable *NoOU* schliesslicht beinhaltet die ID einer „Dummy-OU“. User, welcher keiner OU zugeordnet sind, werden dieser OU zugeordnet. Auch der „Dummy-User“ wird der „Dummy-OU“ zugeordnet. Auch hier ist der Grund derselbe: In der Faktentabelle *FactDim_User* darf es keine leeren Fremdschlüssel geben.

10.3 Analysis Services

Mithilfe der Analyses Services wird der OLAP Cube aufgebaut. Als Datenbasis dienen dabei die Tabellen des DWs, welche regelmässig durch die Packages in den Integrations Services aktualisiert werden.

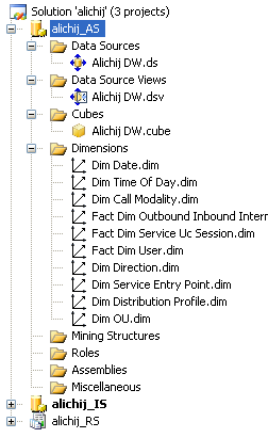


Abbildung 79: Solution alichij, Projekt alichij_AS

10.3.1 Datenquelle (Data Source)

Die Data Source im Projekt alichij_AS spezifiziert den Zugriff auf die Tabellen des DWs Alichij_DW.

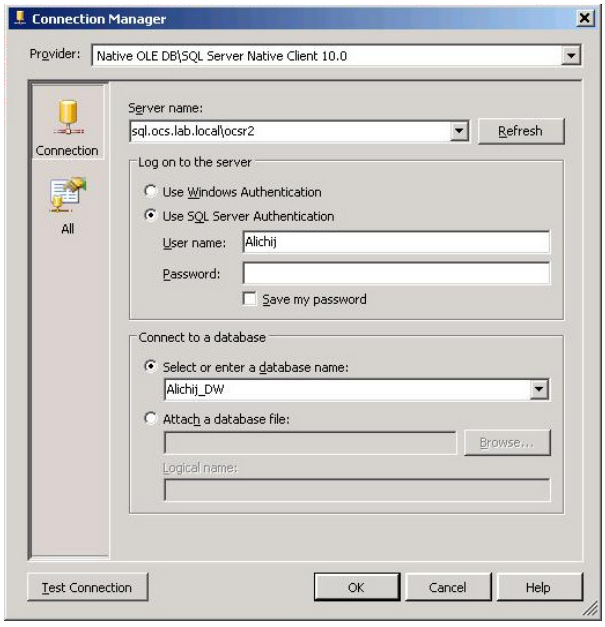


Abbildung 80: Data Source Alichij_DW

10.3.2 Datenquellensicht (Data Source View)

Die Datenquellensicht beinhaltet in unserem Fall alle Tabellen des DWs. Selbstverständlich könnten auch nur Ausschnitte aus einer Quelle übernommen werden. Da in diesem Projekt das DW allerdings auf das Reporting zugeschnitten wurde, sind alle

Tabellen von Relevanz. Die Beziehungen zwischen den Fakten- bzw. Dimensionstabellen werden in der Datenquellensicht definiert.

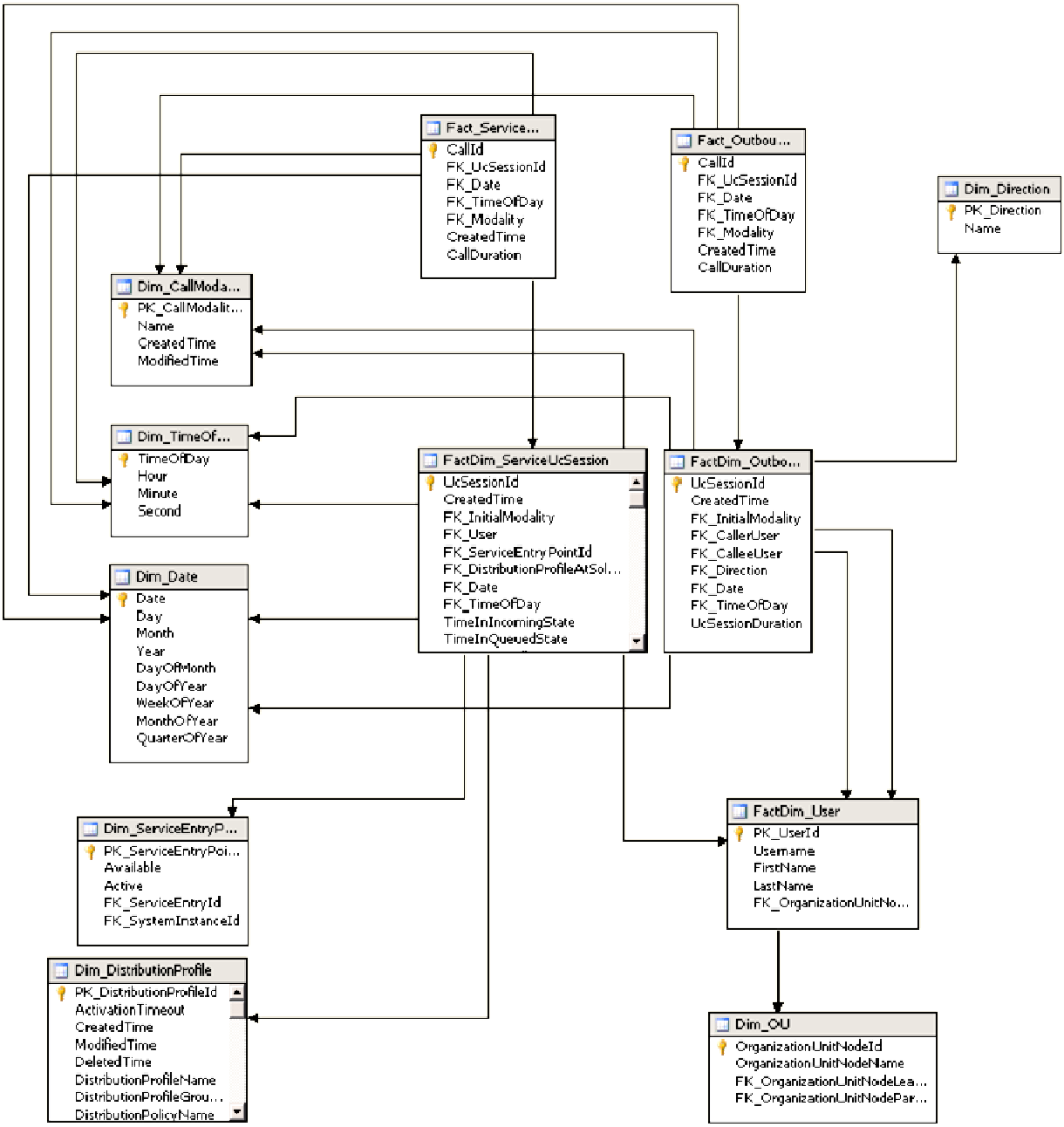


Abbildung 81: Datenquellensicht

10.3.3 Cube

Beim Erstellen des Cubes wird definiert, welche Tabellen der Datenquellensicht als Faktentabellen bzw. Dimensionstabellen fungieren. Weiter kann ausgewählt werden, welche Attribute als Measures in den Cube übernommen werden.

10.3.3.1 Cube Struktur (Cube Structure)

Die folgende Abbildung zeigt die Cubestruktur, die auf der Basis der Datenquellensicht und der Auswahl der Dimensionen bzw. Fakten resultiert. Faktentabellen (auch Faktendimensionstabellen) werden gelb dargestellt, Dimensionstabellen dagegen blau.

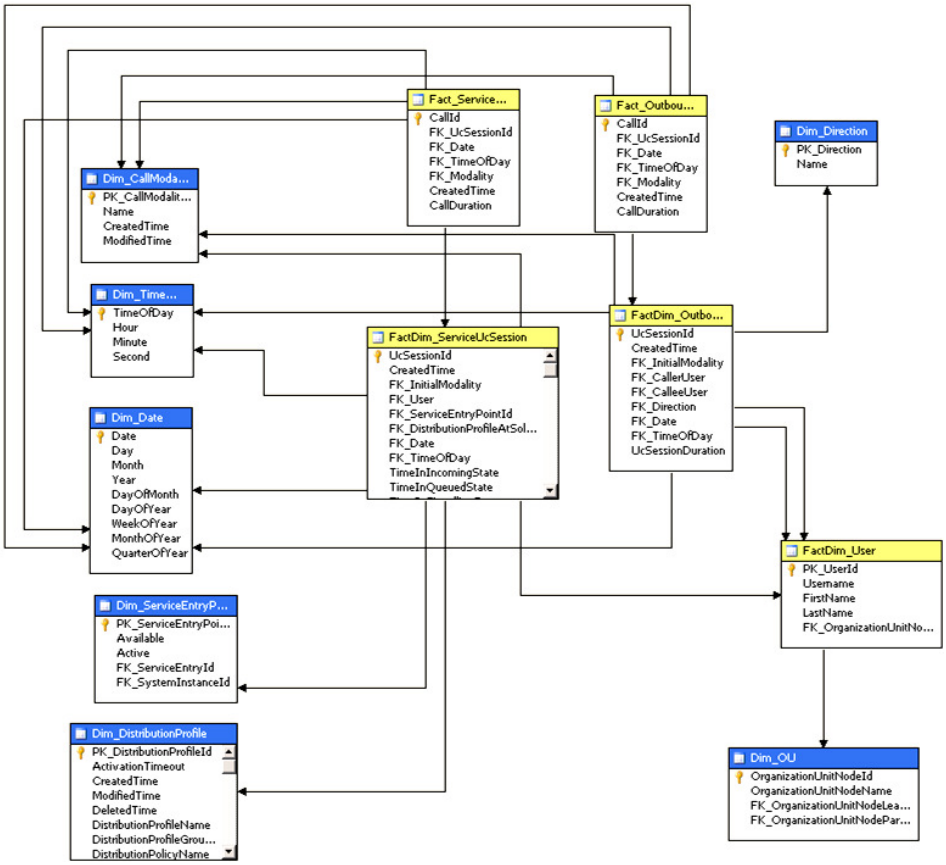
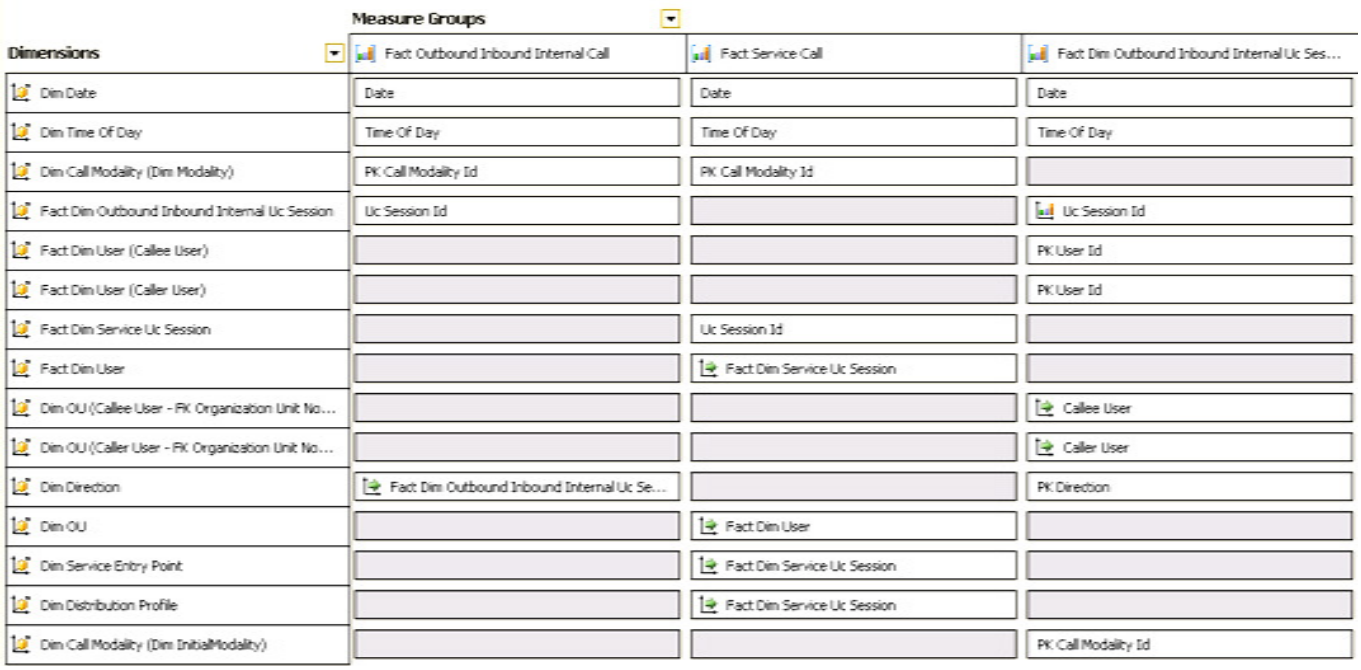


Abbildung 82: Cube Struktur (Data Source View)

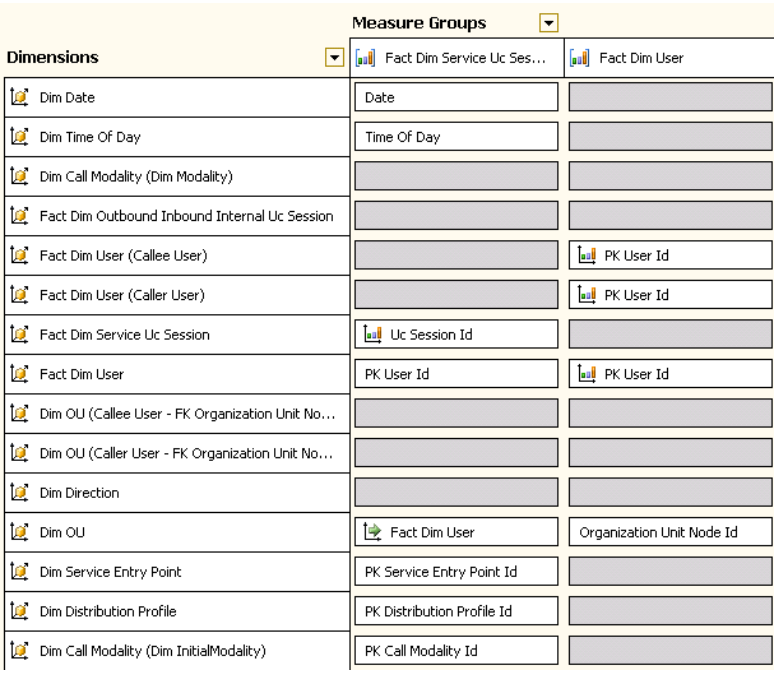
10.3.3.2 Dimensionsverwendung (Dimension Usage)

Dem Cube können Dimensionen hinzugefügt werden. Die Verwendung der Dimensionen durch die verschiedenen Faktentabellen wird in dem Register *Dimensionsverwendung* eingestellt, welches in Visual Studio erscheint, wenn der Cube ausgewählt ist. Die Verwendung der Dimensionen wird in tabellarischer Form angezeigt, wobei am linken Rand die Dimensionen und am oberen Rand die Faktentabellen aufgeführt sind. Die Verwendung einer Dimension für eine bestimmte Faktentabelle wird im Schnittpunkt definiert. Unter anderem muss eingestellt werden, welches Attribut der Faktentabelle der Fremdschlüssel in die Dimensionstabelle ist. Ein Grossteil dieser Tabelle wird beim Erstellen des Cubes automatisch anhand der Beziehungen in der Datenquellensicht generiert.



Dimensions	Measure Groups: Fact Outbound Inbound Internal Call	Fact Service Call	Fact Dim Outbound Inbound Internal Uc Ses…
Dim Date	Date	Date	Date
Dim Time Of Day	Time Of Day	Time Of Day	Time Of Day
Dim Call Modality (Dim Modality)	PK Call Modality Id	PK Call Modality Id	
Fact Dim Outbound Inbound Internal Uc Session	Uc Session Id		Uc Session Id
Fact Dim User (Callee User)			PK User Id
Fact Dim User (Caller User)			PK User Id
Fact Dim Service Uc Session		Uc Session Id	
Fact Dim User		Fact Dim Service Uc Session	
Dim OU (Callee User - FK Organization Unit No…			Callee User
Dim OU (Caller User - FK Organization Unit No…			Caller User
Dim Direction	Fact Dim Outbound Inbound Internal Uc Se…		PK Direction
Dim OU		Fact Dim User	
Dim Service Entry Point		Fact Dim Service Uc Session	
Dim Distribution Profile		Fact Dim Service Uc Session	
Dim Call Modality (Dim InitialModality)			PK Call Modality Id

Abbildung 83: Dimensionsverwendung



Dimensions	Measure Groups: Fact Dim Service Uc Ses…	Fact Dim User
Dim Date	Date	
Dim Time Of Day	Time Of Day	
Dim Call Modality (Dim Modality)		
Fact Dim Outbound Inbound Internal Uc Session		
Fact Dim User (Callee User)		PK User Id
Fact Dim User (Caller User)		PK User Id
Fact Dim Service Uc Session	Uc Session Id	
Fact Dim User	PK User Id	PK User Id
Dim OU (Callee User - FK Organization Unit No…		
Dim OU (Caller User - FK Organization Unit No…		
Dim Direction		
Dim OU	Fact Dim User	Organization Unit Node Id
Dim Service Entry Point	PK Service Entry Point Id	
Dim Distribution Profile	PK Distribution Profile Id	
Dim Call Modality (Dim InitialModality)	PK Call Modality Id	

Abbildung 84: Dimensionsverwendung, Fortsetzung

10.3.4 Dimensionen (Dimensions)

Aus jeder Dimensionstabelle (blau dargestellt in Abbildung 82: Cube Struktur) entsteht eine Cubedimension. Um Drilldowns zu ermöglichen, wurden jedoch auf einigen Dimensionen sogenannte Hierarchien erstellt.

10.3.4.1 Hierarchien (Hierarchies)

Hierarchien werden auf den Dimensionen des Cubes definiert. Folgende Dimensionen besitzen Hierarchien:

- Dim Date
- Dim Time Of Day
- Dim Distribution Profile
- Dim OU

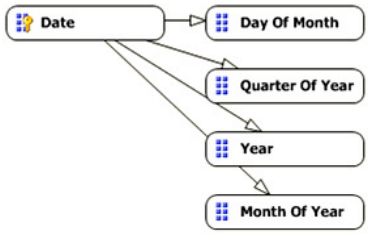
Dimension	Dim Date
Schlüsselattribut der Dimension	Date (Typ Date)
Hierarchie	Year Quarter Of Year Month Of Year Day Of Month
Attributbeziehungen	

Tabelle 30: Dimension Dim Date

Die Dim Date Hierarchie entspricht dem Beispiel in Kapitel 6.1.2.6. Die Attribute Year, QuarterOfYear, MonthOfYear und DayOfMonth aus der Dimensionstabelle Dim_Date werden verwendet, um die Hierarchie aufzubauen. Alle Attribute müssen über die sogenannten Attributbeziehungen (Attribute Relationships) mit dem Schlüsselattribut der Dimension in Beziehung stehen. Diese Beziehungen sind ebenfalls in der obigen Tabelle ersichtlich.

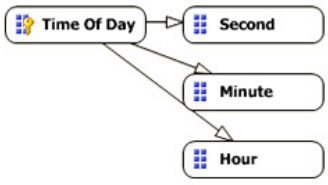
Dimension	Dim Time Of Day
Schlüsselattribut der Dimension	TimeOfDay (Typ Time)
Hierarchie	Hour Minute Second
Attributbeziehungen	

Tabelle 31: Dimension Dim Time Of Day

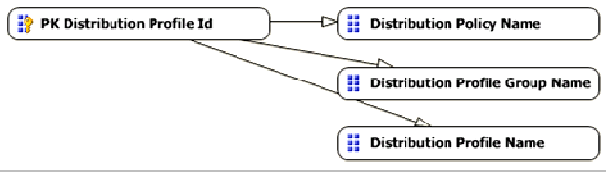
Dimension	Dim Distribution Profile
Schlüsselattribut der Dimension	PK Distribution Profile Id (uniqueidentifier)
Hierarchie	Distribution Policy Name Distribution Profile Group Name Distribution Profile Name
Attributbeziehungen	

Tabelle 32: Dimension Dim Distribution Profile

Dimension	Dim OU
Schlüsselattribut der Dimension	Organization Unit Node Id (uniqueidentifier)
Hierarchie	FK Organization Unit Node Parent Id Organization Unit Node Id

Tabelle 33: Dimension Dim OU

Die Hierarchie der Dimension Dim OU ist keine gewöhnliche Hierarchie, sondern eine sogenannte Parent-Child-Hierarche. Das Schlüsselattribut und somit das Child-Element stellt die Organization Unit Node Id dar. Das zugehörige Parent-Element wird durch den Fremdschlüssel FK_Organization Unit Node Parent Id gefunden. (siehe Kapitel 6.1.2.6.1).

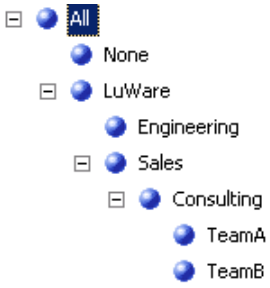


Abbildung 85: OU (Organizational Unit) Hierarchie mit Beispiel OU's

10.4 Reporting Services

Die Reporting Services greifen direkt auf den Cube zu. Anhand der Daten im Cube werden Reports erstellt.

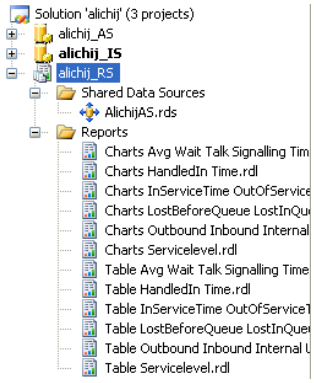


Abbildung 86: Solution alichij, Projekt alichij_RS

10.4.1 Datenquelle

Die Datenquelle *AlichijAS* spezifiziert den Zugriff auf den OLAP Cube *alichij_AS*. Da die SQL Server Analysis Services nur die integrierte Windows Authentifizierung unterstützen und somit von ausserhalb der Domain kein Zugriff möglich wäre, wird auch hier wieder die Datapump für den Zugriff verwendet, welche schon in Kapitel 6.2.1 erwähnt wurde.

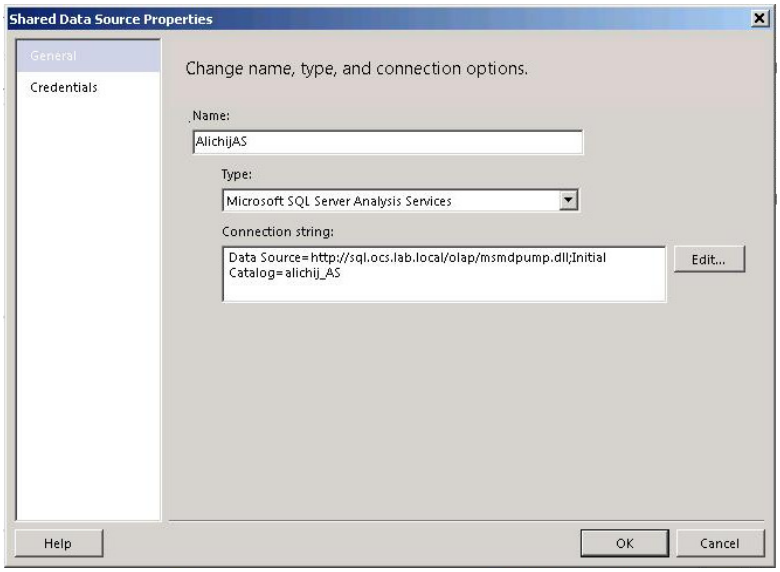


Abbildung 87: Datenquelle AlichijAS

10.4.2 Reports

Anhand der Projektanforderungen wurden 12 Reports erstellt, wobei sechs davon tabellarisch aufgebaut sind, unddie Daten des gesamten Zeitraumes beinhalten sowie Drilldowns erlauben. Die anderen sechs Reports zeigen jeweils vier Diagramme, welche Daten der letzten 7 Tage, 30 Tage, 4 Monate bzw. 2 Jahre anzeigen. Da die tabellarischen Reports Drilldowns erlauben und die Datenmengen sehr gross sind, sind sie nicht optimal, um beispielsweise als PDFs verschickt zu werden, jedoch sind sie über den gesamten Zeitraum vorhanden und bieten daher detailliertere Informationen. Sie sind dazu geeignet, über den Berichtsmanager in einem Webbrowser dargestellt zu werden. Die grafischen Reports, welche jeweils nur einen Teil der Daten anzeigen und nicht parametrisierbar sind, sind hingegen sehr gut dazu geignet, beispielsweise als PDF in regelmässigen Abständen verschickt zu werden.

Allen Reports gemeinsam ist die Aufteilung in ServiceEntryPoints. Sowohl bei den grafischen wie auch bei den tabellarischen Reports wird eine Seite pro ServiceEntryPoint verwendet. Alle Reports basieren auf der gleichen Datenquelle *AlichijAS*, welche auf den Cube *alichij_DW* verweist. Auf diesem Cube werden für jeden Report angepasste MDX-Abfragen abgesetzt, welche dann in einem Berichtsdataset *DataSet1* resultieren.

Die Implementierung der Reports wird anhand je eines Beispiels in tabellarischer und grafischer-Form gezeigt. Die Implementierung der restlichen Diagramme erfolgte in ähnlicher Form.

10.4.2.1 Grafische Reports

Die Namen aller grafischen Reports beginnen mit dem Präfix *Charts* gefolgt von einer möglichst aussagekräftigen Beschreibung der Reports.

Anhand des Reports *Charts Avg Wait Talk Signalling Time* wird beispielhaft gezeigt, wie die Reports enstanden sind. Folgende MDX-Abfrage wird auf dem Cube abgesetzt:

```
WITH MEMBER [Measures].[AvgWaitTime] AS
   AVG(FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,[Measures].[Wait Time]>0),
   [Measures].[Wait Time])

MEMBER [Measures].[AvgTalkTime] AS
   AVG(FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,[Measures].[Talk Time]>0),
   [Measures].[Talk Time])

MEMBER [Measures].[AvgTimeInSignallingState] AS AVG(FILTER([Fact Dim Service Uc Session].
   [Uc Session Id].[All].CHILDREN,[Measures].[Time In Signalling State]>0), [Measures].[Time In
   Signalling State])

MEMBER [Measures].[WaitTime>0_Count] AS
   COUNT(FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,[Measures].[Wait Time]>0))

MEMBER [Measures].[TalkTime>0_Count] AS
   COUNT(FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,[Measures].[Talk Time]>0))
```

```
MEMBER [Measures].[SignallingTime>0_Count] AS
   COUNT(FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,[Measures].
   [Time In Signalling State]>0))

SELECT
   NON EMPTY {[Measures].[AvgWaitTime], [Measures].[AvgTalkTime],
      [Measures].[AvgtimeInSignallingState], [Measures].[WaitTime>0_Count],
      [Measures].[TalkTime>0_Count], [Measures].[SignallingTime>0_Count]} ON COLUMNS,
   NON EMPTY {([Dim Date].[DateHierarchy].[Day Of Month].ALLMEMBERS * [Dim Date].[Date].[Date].ALLMEMBERS
      * [Dim Service Entry Point].[PK Service Entry Point Id].[PK Service Entry Point Id].ALLMEMBERS)}
      DIMENSION PROPERTIES MEMBER_CAPTION, MEMBER_UNIQUE_NAME ON ROWS
FROM [Alichij DW] CELL PROPERTIES VALUE
```

Und das folgende Berichtsdataset entsteht daraus. Wie zu erkennen ist, werden Calculated Members (Kapitel 5.8.3)sowie auch Measures, die an dieser Stelle immer auf der Achse 0 (COLUMNS) angegeben werden, in das Berichtsdataset übernommen. Auch die Dimensionen, die auf der Achse 1 (ROWS) angegeben werden, erscheinen im Berichtsdataset.

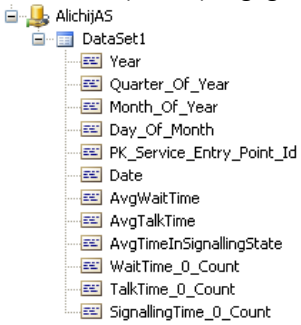


Abbildung 88: Berichtsdataset des Reports *Charts Avg Wait Time Talk Signalling Time*

Bei den grafischen Reports wurden Diagramme verwendet, die verschiedene Felder anbieten, in welche die Felder der Berichtsdatasets gezogen bzw. eigene Ausdrücke eingegeben werden können. Konrekt gibt es bei den verwendeten Diagrammen ein Datenfeld, ein Serienfeld und ein Kategorienfeld.

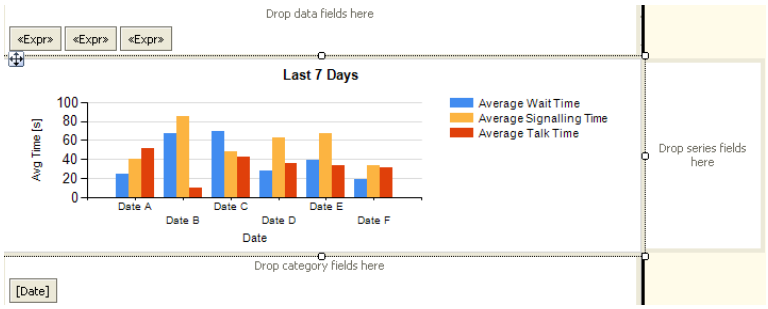


Abbildung 89: Daten-, Kategorien und Serienfelder bei Diagrammen

Die Datenfelder beinhalten die darzustellenden Werte. In diesem Fall handelt es sich um die Calculated Members *Average Wait Time*, *Average Signalling Time* und *Average Talk Time*. Es wäre möglich gewesen, die drei Felder einfach aus dem Berichtsdataset in das Datenfeld des Diagramms zu ziehen. Da die Zahlen aber in Tausendstelsekunden im Cube abgelegt sind, muss noch durch 1000 gerechnet werden, um Sekunden zu erhalten. Deshalb werden im Datenfeld drei Ausdrücke verwendet, die alle analog zum folgenden sind:

```
=Sum(Fields!AvgWaitTime.Value)/1000
```

Im Gegensatz zu den tabellarischen Reports sind die Daten in den grafischen Reports abhängig vom Zeitpunkt, an dem der Report aufgerufen wird. Im Kategorienfeld wird das Feld *Date* aus dem Berichtsdataset verwendet, jedoch muss entsprechend dem aktuellen Datum gefiltert werden, welche Daten angezeigt werden. Um Filter zu definieren, muss im Kontextmenü des Felds *[Date]* im Kategorienfeld *Kategoriegruppeneigenschaften* ausgewählt werden.

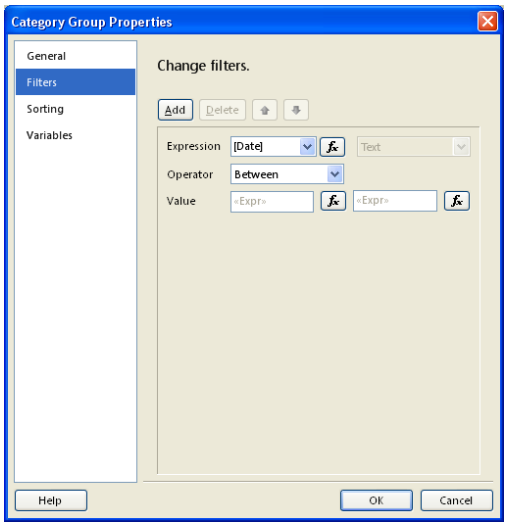


Abbildung 90: Kategoriegruppeneigenschaften von *[Date]*

Der Filter schränkt das Datum mittels zwei Werten ein, die das Anfangs- bzw. Enddatum definieren. Die folgenden beiden Ausdrücke schränken das Datum auf die letzten 7 Tage ein:

Startdatum:

```
=DatePart(DateInterval.Year,DateAdd(DateInterval.Day, -7, Today()))& "-" & Right("0" &
DatePart(DateInterval.Month, DateAdd(DateInterval.Day, -7, Today())), 2) & "-" & Right("0" &
DatePart(DateInterval.Day, DateAdd(DateInterval.Day, -7, Today())), 2)
```

Enddatum:

```
=DatePart(DateInterval.Year,Today())& "-" & DatePart(DateInterval.Month, Today()) & "-" &
DatePart(DateInterval.Day, Today())
```

10.4.2.2 Tabellarische Reports

Die Namen der tabellarischen Reports beginnen alle mit dem Präfix *Table* gefolgt von einem möglichst aussagekräftigen Namen.

Wie bei den grafischen Reports wird auch bei den tabellarischen eine MDX-Abfrage auf dem Cube abgesetzt und ein Berichtsdataset generiert. Dieses wird dann aber für eine tabellarische Darstellung verwendet. Man kann die Tabellen im Reportdesigner von Hand erstellen, viel einfacher ist es jedoch mit dem Assistenten, da dieser sowieso eine tabellarische Darstellung wählt. Der Assistent verlangt nach einem MDX-Statement. Anhand dieses Statements wird die Tabelle dann erstellt.

Anhand des Reports *Table Servicelevel* wird die Implementierung der tabellarischen Reports beispielhaft gezeigt.

Folgende MDX-Abfrage

```
WITH MEMBER [HandledWithinTimeParam] AS
   20000

MEMBER [AbandonedWithinTimeParam] AS
   5000

MEMBER [Measures].[HandledWithin<=HandledWithinTimeParam] AS
   COUNT(NONEMPTY((FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,
   [Measures].[Handled In Time]>0 AND [Measures].[Handled In Time]<=[HandledWithinTimeParam])),
   [Measures].[Fact Dim Service Uc Session Count]))

MEMBER [Measures].[AbandonedWithin<=AbandonedWithinTimeParam] AS
   COUNT(NONEMPTY((FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,
   [Measures].[Abandoned In Time]<=[AbandonedWithinTimeParam] AND
   [Measures].[Time In Established State]>0)),[Measures].[Fact Dim Service Uc Session Count]))

MEMBER [Measures].[OutOfServiceTime] AS
   COUNT(NONEMPTY((FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,
   [Measures].[Is Out Of Service]=TRUE)),[Measures].[Fact Dim Service Uc Session Count]))
```

```
MEMBER [Measures].[LostBeforeQueue] AS
   COUNT(NONEMPTY((FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,
   [Measures].[Time In Queued State]=0)),[Measures].[Fact Dim Service Uc Session Count]))

MEMBER [Measures].[Servicelevel] AS
   [Measures].[HandledWithin<=HandledWithinTimeParam]
   / ([Measures].[Fact Dim Service Uc Session Count]-[Measures].[OutOfServiceTime]
   - [Measures].[LostBeforeQueue] - [Measures].[AbandonedWithin<=AbandonedWithinTimeParam])

SELECT
   NON EMPTY {[Measures].[Servicelevel], [Measures].[HandledWithin<=HandledWithinTimeParam],
      [Measures].[AbandonedWithin<=AbandonedWithinTimeParam], [Measures].[OutOfServiceTime],
      [Measures].[LostBeforeQueue], [Measures].[Fact Dim Service Uc Session Count]} ON COLUMNS,
   NON EMPTY {([Dim Date].[DateHierarchy].[Day Of Month].ALLMEMBERS * [Dim Date].[Date].[Date].ALLMEMBERS
      * [Dim Service Entry Point].[PK Service Entry Point Id].[PK Service Entry Point Id].ALLMEMBERS)}
      DIMENSION PROPERTIES MEMBER_CAPTION, MEMBER_UNIQUE_NAME ON ROWS
FROM [Alichij DW] CELL PROPERTIES VALUE
```

ergibt folgendes Berichtsdataset

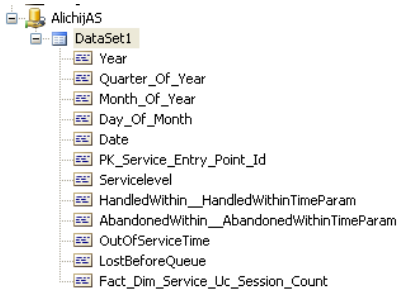


Abbildung 91: : Berichtsdataset des Reports *Table Servicelevel*

Eine Schwierigkeit gerade bei diesem Report stellte die Aggregation der Daten dar. Der Servicelevel wird standardmässig nur für die unterste Ebene der Hierarchie berechnet und für die weiteren Hierarchien aggregiert. Beim Servicelevel jedoch gibt es keine einfache Funktion wie zum Beispiel die Durchschnittsfunktion oder die Summe, welche die Daten korrekt aggregiert. Denn der Durchschnitt der Servicelevel der einzelnen Tage ist NICHT gleich dem berechneten Servicelevel des Monats. Diese Tatsache führt dazu, dass die Daten nicht aggregiert werden dürfen, sondern auf jeder Stufe erneut berechnet werden müssen. Da für die Berechnung des Servicelevel aber noch weitere Daten benötigt werden, mussten diese ebenfalls mittels der MDX-Abfrage in das Berichtsdataset kopiert werden, wie in der obigen Abbildung ersichtlich ist.
Für die Neuberechnung des Servicelevels wird auf allen Ebenen der Tabelle, ausser auf der untersten, wo der Servicelevel ohnehin schon richtig berechnet wird, folgender Ausdruck verwendet:

```
=Round(Sum(Fields!HandledWithin__HandledWithinTimeParam.Value)
   / (Sum(Fields!Fact_Dim_Service_Uc_Session_Count.Value - Fields!OutOfServiceTime.Value
   - Fields!LostBeforeQueue.Value - Fields!AbandonedWithin__AbandonedWithinTimeParam.Value)),2) * 100
```

11 Implementierung LUBIWebFrontend

11.1 AnalysisServicesQueryService

Der AnalysisServicesQueryService ist ein Webservice und stellt die Schnittstelle zwischen der Silverlight-Applikation und den Analysis Services auf dem SQL-Server dar. Die Implementation dieses Webservice wurde, wie bereits erwähnt, nötig, da in einer Silverlight-Applikation ADOMD.NET nicht zur Verfügung steht. Auf die Verwendung von ADOMD.NET zu verzichten und die Commands als XMLA/SOAP-Message selber an die SSAS zu senden, wurde wegen dem grossen Aufwand und der Fehleranfälligkeit nie in Betracht gezogen.

11.1.1 Ablauf einer GetData()-Methode

1. AdomdConnection erstellen
2. AdomdConnection öffnen
3. AdomdCommand auf der Connection erstellen
4. CommmandText setzen
5. CommandParameter setzen
6. Command ausführen und Resultat in CellSet (In-Memory-Abbildung des Resultat-Cubes) ablegen
7. CellSet prozessieren und Daten extrahieren
8. Resultat zurückgeben

Code-Beispiel:

```
using (AdomdConnection conn = new AdomdConnection(connectionString))
  {
  try
    {
      conn.Open();
      AdomdCommand cmd = conn.CreateCommand();

      string mdxQuery = @"WITH SET [Modalities] AS
        [Dim InitialModality].[Name].[ALL].CHILDREN
      MEMBER [Measures].[Anteil] AS
        [Measures].[Fact Dim Service Uc Session Count]
        / ([Measures].[Fact Dim Service Uc Session Count], Root([Dim InitialModality]))

      SELECT
        {[Measures].[Fact Dim Service Uc Session Count], [Measures].[Anteil]} ON 0,
        [Modalities] ON 1
      FROM [Alichij DW]
      WHERE (StrToMember(@PeriodStart):StrToMember(@PeriodEnd),StrToSet(@ServiceEntryPoints))";

      cmd.CommandText = mdxQuery;
      cmd.Parameters.Add("PeriodStart", "[Dim Date].[Date].&[" +
        String.Format("{0:yyyy-MM-ddTHH:mm:ss}", periodStart) + "]");
      cmd.Parameters.Add("PeriodEnd", "[Dim Date].[Date].&[" +
        String.Format("{0:yyyy-MM-ddTHH:mm:ss}", periodEnd) + "]");
      cmd.Parameters.Add("ServiceEntryPoints",
        CreateServiceEntryPointsMDX(serviceEntryPoints));
      CellSet cs = cmd.ExecuteCellSet();
      // Prozessieren des CellSets
      ...
```

Folgende Punkte gilt es zu beachten:

- Gesetzte Parameter müssen, sofern sie einen Member oder Set in der MDX-Abfrage repräsentieren, durch eine enstprechende MDX-Funktion (StrToMember, StrToSet) zu einem solchen gemacht werden. Ansonsten resultieren Parse-Errors.
- Die Formate, insbesondere das Datumsformat, müssen exakt eingehalten werden. Ansonsten resultieren ebenfalls Parse-Errors auf der SSAS-Seite.

11.1.2 Konfigurationsdaten in Web.xml

Der ConnectionString sowie einige wenige Thresholds die relevant für MDX-Abfragen sind, lassen sich im Web.xml konfigurieren.

Name	Beschreibung	Default-Wert
LUBIAnalysisServicesHTTPPump	Beinhaltet den ConnectionString für die Connection auf die msmdpump.dll	`Provider=MSOLAP; Data Source=http://sql.ocs.lab.local/olap/msmdpump.dll; Catalog=alichij_AS;`
handlingTimeShortThreshold	Erlaubt das Setzen eines Thresholds für MDX-Abfragen beim Zählen von Conversations welche innerhalb einer bestimmten Zeit behandelt wurden. Zeit in Sekunden	20
handlingTimeNormalThreshold	Erlaubt das Setzen eines zweiten Thresholds für MDX-Abfragen beim Zählen von Conversations welche innerhalb einer bestimmten Zeit behandelt wurden. Zeit in Sekunden	60
handlingTimeThresholdForServiceLevel	Erlaubt das Setzten eines Threshold für die Berechnung des Servicelevels in der MDX-Query. Hier kann festgelegt werden, welche Conversations als HandledConversationsWithin... gezählt werden. Zeitangabe in Sekunden.	20
abandonedTimeThresholdForServiceLevel	Erlaubt das Setzen eines Threshold ebenfalls für die Berechnung des Servicelevels. Hier wird festgelegt, ab welcher Zeit Conversations als Abandoned gezählt werden. Zeitangabe in Sekunden.	5

Tabelle 34: Konfigurationsmöglichkeiten Web.config

Diese Daten können folgendermassen ausgelesen und anschliessend im Webservice verwendet werden:

- Für den ConnectionString:

```
ConfigurationManager.ConnectionStrings["LUBIAnalysisServicesHTTPPump"].ConnectionString;
```

- Für die Thresholds:

```
Int32.Parse(ConfigurationManager.AppSettings.Get("handlingTimeShortThreshold"));
```

11.1.3 MDX-Queries

Dieses Kapitel beschreibt die erstellten MDX-Queries inkl. der verwendeten Parameter welche im Webservice für die Abfrage der Daten aus dem OLAP-Cube verwendet werden.

11.1.3.1 Acceptance Time Agent

```
WITH SET [User] AS
   FILTER([Fact Dim User].[Username].[All].CHILDREN, ([Measures].[Fact Dim User Count],
   [Dim OU].[FK Organization Unit Node Parent Id].&[{182E6376-F075-47AD-9635-F4F98D0546CD}]))

MEMBER [Measures].[MaxTimeInSignallingState] AS
   MAX([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,
   [Measures].[Time In Signalling State])
```

```
MEMBER [Measures].[AvgTimeInSignallingState] AS
   AVG([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,
   [Measures].[Time In Signalling State])

SELECT {[Measures].[MaxTimeInSignallingState], [Measures].[AvgTimeInSignallingState]} ON 0,
[User] ON 1
FROM [Alichij DW]
WHERE (StrToMember(@PeriodStart) : StrToMember(@PeriodEnd), StrToSet(@ServiceEntryPoints))
```

Zuerst wird ein Set mit allen relevanten Kundenberatern erstellt. Dazu wird die Dimension User gefiltert und nur diejenigen die der obersten OU-Ebene angehören, zum Set hinzugefügt. Sobald eine Authentisierungs-Lösung entwickelt wird, muss dieser Wert selbstverständlich auch parametrisierbar sein, da ansonsten jeweils immer alle Zahlen für jeden Kundenberater der Firma aus dem Cube bezogen werden. Anschliessend wird der Maximal- und der Durschnitts-Wert der Annahmezeit berechnet.

Im Select-Statement werden die vorher berechneten Werte auf der Achse 0 und die Benutzer, zu denen die Werte gehören auf der Achse 1 selektiert.

Parametrisieren lässt sich die Abfrage im Bezug auf die Zeitspanne, für welche die Werte berechnet werden und auf die zu berücksichtigenden ServiceEntryPoints.

11.1.3.2 Conversations Over Day

```
WITH SET [TimeOfDay] AS
   {[Dim Time Of Day].[Hour].MEMBERS}

SET [Time] AS
   StrToMember(@PeriodStart) : StrToMember(@PeriodEnd)

MEMBER [Measures].[Uc Session Count Avg] AS
   Avg([Time], [Measures].[Fact Dim Service Uc Session Count])

SELECT [Measures].[Uc Session Count Avg] ON 0,
   [TimeOfDay] ON 1
FROM [Alichij DW]
WHERE (StrToMember(@PeriodStart) : StrToMember(@PeriodEnd), StrToSet(@ServiceEntryPoints))
```

Zuerst wird ein Set der Stunden eines Tages erstellt. Anschliessend wird die Zeitperiode festgelegt, für welche die durchschnittliche Verteilung der Konversationen über einen Tag berechnet wird. Im MEMBER [Measures].[Uc Session Count Avg] wird die durchschnittliche Anzahl Conversations berechnet.

Auf Achse 0 werden die berechneten Werte selektiert und auf der Achse 1 die Stunden eines Tages. Einschränken kann man das Ergebnis mit den zu berücksichtigenden ServiceEntryPoints und mit der Zeitspanne, für welche die Werte berechnet werden.

11.1.3.3 Conversations per Agent

```
WITH SET [User] AS
   FILTER([Fact Dim User].[Username].[All].CHILDREN, ([Measures].[Fact Dim User Count],
   [Dim OU].[FK Organization Unit Node Parent Id].&[{182E6376-F075-47AD-9635-F4F98D0546CD}]))

MEMBER [Measures].[HandledIn<=ShortThreshold] AS
   COUNT(NONEMPTY((FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,
   [Measures].[Handled In Time]<=@HandlingTimeShort)),[Measures].[Fact Dim Service Uc Session Count]))

MEMBER [Measures].[ShortThreshold<HandledIn<=NormalThreshold] AS
   COUNT(NONEMPTY((FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,
   [Measures].[Handled In Time]>@HandlingTimeShort
   AND [Measures].[Handled In Time]<=@HandlingTimeNormal)),
   [Measures].[Fact Dim Service Uc Session Count]))

MEMBER [Measures].[NormalThreshold<HandledIn] AS
   COUNT(NONEMPTY((FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,
  [Measures].[Handled In Time]>@HandlingTimeNormal)),
  [Measures].[Fact Dim Service Uc Session Count]))

SELECT {[Measures].[HandledIn<=ShortThreshold], [Measures].[ShortThreshold<HandledIn<=NormalThreshold],
   [Measures].[NormalThreshold<HandledIn]} ON 0,
[User] ON 1
FROM [Alichij DW]
WHERE ((StrToMember(@PeriodStart) : StrToMember(@PeriodEnd)), StrToSet(@ServiceEntryPoints))
```

Zuerst wird wieder ein Set der relevanten User erstellt. Auch hier gilt wieder, dass dies, sobald eine Authentifizierungs-Lösung entwickelt wurde, parametrisierbar sein muss. Anschliessend werden die Werte für die Conversationen berechnet, welche innerhalb bestimmter Thresholds behandelt wurden. Diese Thresholds lassen sich über das Web.config anpasse (siehe Kapitel 11.1.2).

Auf der Achse 0 werden die berechneten Werte selektiert und auf Achse 1 die Kundenberater.

11.1.3.4 Conversations per Agent/ServiceEntryPoint

```
WITH SET [User] AS
   FILTER([Fact Dim User].[Username].[All].CHILDREN,
   ([Measures].[Fact Dim User Count],
   [Dim OU].[FK Organization Unit Node Parent Id].&[{182E6376-F075-47AD-9635-F4F98D0546CD}]))

SET [ServiceEntryPoints] AS
   StrToSet(@ServiceEntryPoints)

SELECT [ServiceEntryPoints] ON 0,
[User] ON 1
FROM [Alichij DW]
WHERE ([Measures].[Fact Dim Service Uc Session Count],
   (StrToMember(@PeriodStart) : StrToMember(@PeriodEnd)))
```

Nachdem nach bekanntem Muster ein Set der Kundenberater erstellt wurde, werden die zu berücksichtigenden ServiceEntryPoint als Set erstellt.

Auf Achse 0 werden die Service Entry Points selektiert und auf Achse 1 die User. Das zu berücksichtigende Measure wird in der WHERE-Klausel festgelegt, wo auch wieder die bekannte Einschränkung für eine Zeitspanne gemacht werden kann.

11.1.3.5 Initial Modalities

```
WITH SET [Modalities] AS
   [Dim InitialModality].[Name].[ALL].CHILDREN

MEMBER [Measures].[Anteil] AS
   [Measures].[Fact Dim Service Uc Session Count] /
   ([Measures].[Fact Dim Service Uc Session Count], Root([Dim InitialModality]))

SELECT {[Measures].[Fact Dim Service Uc Session Count], [Measures].[Anteil]} ON 0,
[Modalities] ON 1
FROM [Alichij DW]
WHERE (StrToMember(@PeriodStart) : StrToMember(@PeriodEnd), StrToSet(@ServiceEntryPoints))
```

Nachdem ein Set der Namen der vorhandenen Initial-Modalitäten erstellt wurde, wird der prozentuale Anteil der verwendeten einzelnen Modalitäten an der Gesamtzahl der Conversations berechnet.

Selektiert werden die berechneten Werte auf der Achse 0, die Namen der Modalitäten auf der Achse 1. Die Parametrisierungsmöglichkeiten sind aus den vorangehenden Beschreibungen bekannt.

11.1.3.6 Non Service Conversations

```
WITH SET [User] AS
   FILTER([Caller User].[Username].[All].CHILDREN,
   ([Measures].[Fact Dim User Count],
   [Dim OU].[FK Organization Unit Node Parent Id].&[{182E6376-F075-47AD-9635-F4F98D0546CD}]))

SELECT {[Dim Direction].[Name].&[Inbound], [Dim Direction].[Name].&[Outbound],
   [Dim Direction].[Name].&[Internal]} ON 0,
[User] ON 1
FROM [Alichij DW]
WHERE ([Measures].[Fact Dim Outbound Inbound Internal Uc Session Count],
   (StrToMember(@PeriodStart) : StrToMember(@PeriodEnd)))
```

Nachdem nach bekanntem Muster ein Set der Kundenberater erstellt wurde, werden auf der Achse 0 die aufsummierten (als Standard im Cube eingerichtet, deshalb kein SUM nötig) Gesamtzahlen der Conversations selektiert. Auf Achse 1 werden die Kundenberater ausgewählt. Parametrisierungsmöglichkeiten sind dieselben wie in den vorangegangenen MDX-Queries.

11.1.3.7 Talk Time Non Service Conversations

```
WITH SET [CallerUser] AS
   FILTER([Caller User].[Username].[All].CHILDREN, ([Measures].[Fact Dim User Count],
   [Dim OU].[FK Organization Unit Node Parent Id].&[{182E6376-F075-47AD-9635-F4F98D0546CD}]))
```

```
MEMBER [Measures].[OutboundMax] AS
   MAX(([Fact Dim Outbound Inbound Internal Uc Session].[Uc Session Id].[All].CHILDREN,
   [Dim Direction].[Name].&[Outbound]), [Measures].[Uc Session Duration])

MEMBER [Measures].[OutboundAvg] AS
   AVG(([Fact Dim Outbound Inbound Internal Uc Session].[Uc Session Id].[All].CHILDREN,
   [Dim Direction].[Name].&[Outbound]), [Measures].[Uc Session Duration])

MEMBER [Measures].[InboundMax] AS
   MAX(([Fact Dim Outbound Inbound Internal Uc Session].[Uc Session Id].[All].CHILDREN,
   [Dim Direction].[Name].&[Inbound]), [Measures].[Uc Session Duration])

MEMBER [Measures].[InboundAvg] AS
   AVG(([Fact Dim Outbound Inbound Internal Uc Session].[Uc Session Id].[All].CHILDREN,
   [Dim Direction].[Name].&[Inbound]), [Measures].[Uc Session Duration])

MEMBER [Measures].[InternalMax] AS
   MAX(([Fact Dim Outbound Inbound Internal Uc Session].[Uc Session Id].[All].CHILDREN,
   [Dim Direction].[Name].&[Internal]), [Measures].[Uc Session Duration])

MEMBER [Measures].[InternalAvg] AS
   AVG(([Fact Dim Outbound Inbound Internal Uc Session].[Uc Session Id].[All].CHILDREN,
   [Dim Direction].[Name].&[Internal]), [Measures].[Uc Session Duration])

SELECT {[Measures].[OutboundMax], [Measures].[OutboundAvg], [Measures].[InboundMax],
   [Measures].[InboundAvg], [Measures].[InternalMax], [Measures].[InternalAvg]} ON 0,
[CallerUser] ON 1
FROM [Alichij DW]
WHERE ((StrToMember(@PeriodStart) : StrToMember(@PeriodEnd)))
```

Zuerst wird ein Set mit den Initianten von Non-Service Conversations erstellt. Anschliessend werden die verschiedenen Werte der einzelnen Kategorien berechnet.

Die Werte werden auf der Achse 0 selektiert, während Initianten der Conversations auf der Achse 1 aufgelistet werden. Einschränken lässt sich das Resultat mit der Zeitdauer, welche für die Berechnung der Resultate berücksichtigt werden soll. Eine Einschränkung nach Service Entry Points macht bei Non-Service Conversations keinen Sinn und ist deshalb nicht möglich.

11.1.3.8 Queue Time

```
WITH MEMBER [Measures].[MinWaitTime] AS
   MIN(FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN, [Measures].[Wait Time]>0),
   [Measures].[Wait Time])

MEMBER [Measures].[MaxWaitTime] AS
   MAX(FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,[Measures].[Wait Time]>0),
   [Measures].[Wait Time])

MEMBER [Measures].[AvgWaitTime] AS
   AVG(FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,[Measures].[Wait Time]>0),
   [Measures].[Wait Time])

SET [ServiceEntryPoints] AS
   StrToSet(@ServiceEntryPoints)

SELECT {[Measures].[MinWaitTime], [Measures].[MaxWaitTime], [Measures].[AvgWaitTime]} ON 0,
[ServiceEntryPoints] ON 1
FROM [Alichij DW]
WHERE ((StrToMember(@PeriodStart) : StrToMember(@PeriodEnd)))
```

Zuerst werden wieder die Werte berechnet und anschliessend auf der Achse 0 selektiert. Auf der Achse 1 werden die Service Entry Points aufgeführt. Parametrisieren lässt sich diese MDX-Query einerseits mit der zu berücksichtigenden Zeitspanne und mit den zu berücksichtigenden Service Entry Points.

11.1.3.9 Servicelevel

```
WITH SET [Time] AS
   (StrToMember(@PeriodStart) : StrToMember(@PeriodEnd))

SET [ServiceEntryPoints] AS
   (StrToSet(@ServiceEntryPoints))

MEMBER [HandledWithinTimeParam] AS
   @HandledWithinTimeParam

MEMBER [AbandonedWithinTimeParam] AS
   @AbandonedWithinTimeParam
```

```
MEMBER [Measures].[HandledWithin<=HandledWithinTimeParam] AS
   COUNT(NONEMPTY((FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,
   [Measures].[Handled In Time]>0 AND [Measures].[Handled In Time]<=[HandledWithinTimeParam])),
   [Measures].[Fact Dim Service Uc Session Count]))

MEMBER [Measures].[AbandonedWithin<=AbandonedWithinTimeParam] AS
   COUNT(NONEMPTY((FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,
   [Measures].[Abandoned In Time]<=[AbandonedWithinTimeParam] AND
   [Measures].[Time In Established State]>0)),[Measures].[Fact Dim Service Uc Session Count]))

MEMBER [Measures].[OutOfServiceTime] AS
   COUNT(NONEMPTY((FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,
   [Measures].[Is Out Of Service]=TRUE)),[Measures].[Fact Dim Service Uc Session Count]))

MEMBER [Measures].[LostBeforeQueue] AS
   COUNT(NONEMPTY((FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,
   [Measures].[Time In Queued State]=0)),[Measures].[Fact Dim Service Uc Session Count]))

MEMBER [Measures].[SL] AS
   [Measures].[HandledWithin<=HandledWithinTimeParam] / ([Measures].[Fact Dim Service Uc Session Count]
   - [Measures].[OutOfServiceTime] - [Measures].[LostBeforeQueue]
   - [Measures].[AbandonedWithin<=AbandonedWithinTimeParam])

MEMBER [Measures].[Servicelevel] AS
   Iif([Measures].[SL] =-1, 0, [Measures].[SL])

SELECT [ServiceEntryPoints] ON 0,
[Time] ON 1
FROM [Alichij DW]
WHERE [Measures].[Servicelevel]
```

Zuerst werden die beiden Sets mit der Zeitspanne und der Service Entry Points erstellt. Anschliessend werden die Parameter für die Thresholds (Kapitel 11.1.2) übernommen. Nun werden die einzelnen Werte berechnet. Dabei werden die zuvor gesetzten Thresholds berücksichtigt.

Auf der Achse 0 werden die Service Entry Points selektiert und auf der Achse 1 werden die Daten der Zeitspanne aufgelistet. Welches Measure berücksichtigt wird, legt die WHERE-Klausel fest.

11.1.3.10 Talk Time Agent

```
WITH SET [User] AS
   FILTER([Fact Dim User].[Username].[All].CHILDREN, ([Measures].[Fact Dim User Count],
   [Dim OU].[FK Organization Unit Node Parent Id].&[{182E6376-F075-47AD-9635-F4F98D0546CD}]))

MEMBER [Measures].[MaxTalkTime] AS
   MAX([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN, [Measures].[Talk Time])

MEMBER [Measures].[AvgTalkTime] AS
   AVG([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN, [Measures].[Talk Time])

SELECT {[Measures].[MaxTalkTime], [Measures].[AvgTalkTime]} ON 0,
[User] ON 1
FROM [Alichij DW]
WHERE ((StrToMember(@PeriodStart) : StrToMember(@PeriodEnd)), StrToSet(@ServiceEntryPoints))
```

Nach dem Erstellen des Sets mit den betroffen Kundenberatern werden die Werte berechnet. Diese berechneten Werte werden auf der Achse 0 aufgelistet, während auf der Achse 1 die Kundenberater selektiert werden.

Parameter, die gesetzt werden können, umfassen die Zeitspanne und die Service Entry Points.

11.1.3.11 Team Conversations

```
WITH SET [Time] AS
   (StrToMember(@PeriodStart) : StrToMember(@PeriodEnd))

MEMBER [Measures].[HandledIn<=ShortThreshold] AS
   COUNT(NONEMPTY((FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,
   [Measures].[Handled In Time]<=@HandlingTimeShort AND [Measures].[Handled In Time]>0)),
   [Measures].[Fact Dim Service Uc Session Count]))

MEMBER [Measures].[ShortThreshold<HandledIn<=NormalThreshold] AS
   COUNT(NONEMPTY((FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,
   [Measures].[Handled In Time]>@HandlingTimeShort AND
   [Measures].[Handled In Time]<=@HandlingTimeNormal)),
   [Measures].[Fact Dim Service Uc Session Count]))
```

```
MEMBER [Measures].[NormalThreshold<HandledIn] AS
   COUNT(NONEMPTY((FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,
   [Measures].[Handled In Time]>@HandlingTimeNormal)),[Measures].[Fact Dim Service Uc Session Count]))

MEMBER [Measures].[LostBeforeQueue] AS
   COUNT(NONEMPTY((FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,
   [Measures].[Time In Queued State]=0)),[Measures].[Fact Dim Service Uc Session Count]))

MEMBER [Measures].[LostInQueue] AS
   COUNT(NONEMPTY((FILTER([Fact Dim Service Uc Session].[Uc Session Id].[All].CHILDREN,
   [Measures].[Time In Queued State]>0 AND [Measures].[Time In Signalling State]=0)),
   [Measures].[Fact Dim Service Uc Session Count]))

SELECT {[Measures].[HandledIn<=ShortThreshold],[Measures].[ShortThreshold<HandledIn<=NormalThreshold],
   [Measures].[NormalThreshold<HandledIn], [Measures].[LostBeforeQueue],[Measures].[LostInQueue]} ON 0,
[Time] ON 1
FROM [Alichij DW]
WHERE (StrToSet(@ServiceEntryPoints))
```

Es wird ein Set mit den Daten der Zeitspanne erstellt. Anschliessend werden die Werte berechnet. Dabei müssen diverse Werte, die das Ergebnis verfälschen würden, ausgefiltert werden. Dies sind z.B. Werte von Conversations, die vor dem Eintreffen in der Queue terminiert werden.

Auf der Achse 0 werden wiederum die berechneten Werte selektiert, während auf der Achse 1 die Daten der Zeitspanne aufgeführt werden. Parameter können für die Service Entry Points und die zu berücksichtigende Zeitspanne gesetzt werden.

11.1.4 Zugriff auf Metadaten des Cubes[29]

Möchte man auf die Metadaten des Resultats einer MDX-Abfrage zugreifen, bietet das CellSet eine grosse Unterstützung. Im Folgenden wird aufgezeigt, wie im AnalysisServicesQueryService genutzt wird.

Folgender Code-Abschnitt zeigt, wie im AnalysisServicesQueryService die Member der Service Entry Point-Dimension für die Anzeige in der Combobox auf dem UI ausgelesen werden.

```
using (AdomdConnection conn = new AdomdConnection(connectionString))
{
  try
  {
    conn.Open();
    Level l = conn.Cubes["Alichij DW"].Dimensions["Dim Service Entry Point"].
      Hierarchies["Pk Service Entry Point Id"].Levels["Pk Service Entry Point Id"];
    MemberCollection mc = l.GetMembers();
    foreach (Member m in mc)
    {
      ServiceEntryPoint se = new ServiceEntryPoint();
      se.Guid = m.Name;
      se.UniqueName = m.UniqueName;
      results.Add(se);
    }
...
```

11.2 Graphical User Interface

Das GUI der Applikation wurde, wie im Kapitel 9.4**Error! Reference source not found.** beschrieben, mittels der Prism-Bibliothek von Microsoft umgesetzt. Für die Visualisierung der Daten in ansprechenden Charts konnten die Silverlight Controls der Firma Telerik verwendet werden. Das INS hat sich bereits vor dem Beginn der Bachelorarbeit diese Bibliothek gekauft. In den folgenden Unterkapiteln möchte ich auf einige Punkte aus den Bereichen Prism und Telerik-Controls eingehen.

11.2.1 Prism

11.2.1.1 Umsetzung Regions

Als Ergänzung für die Beschreibung des Region-Designs in Kapitel 9.4.1 soll folgende Darstellung dienen:

[29] Quelle: (Gorbach, Melomed, Berger, & Bateman, 2006)

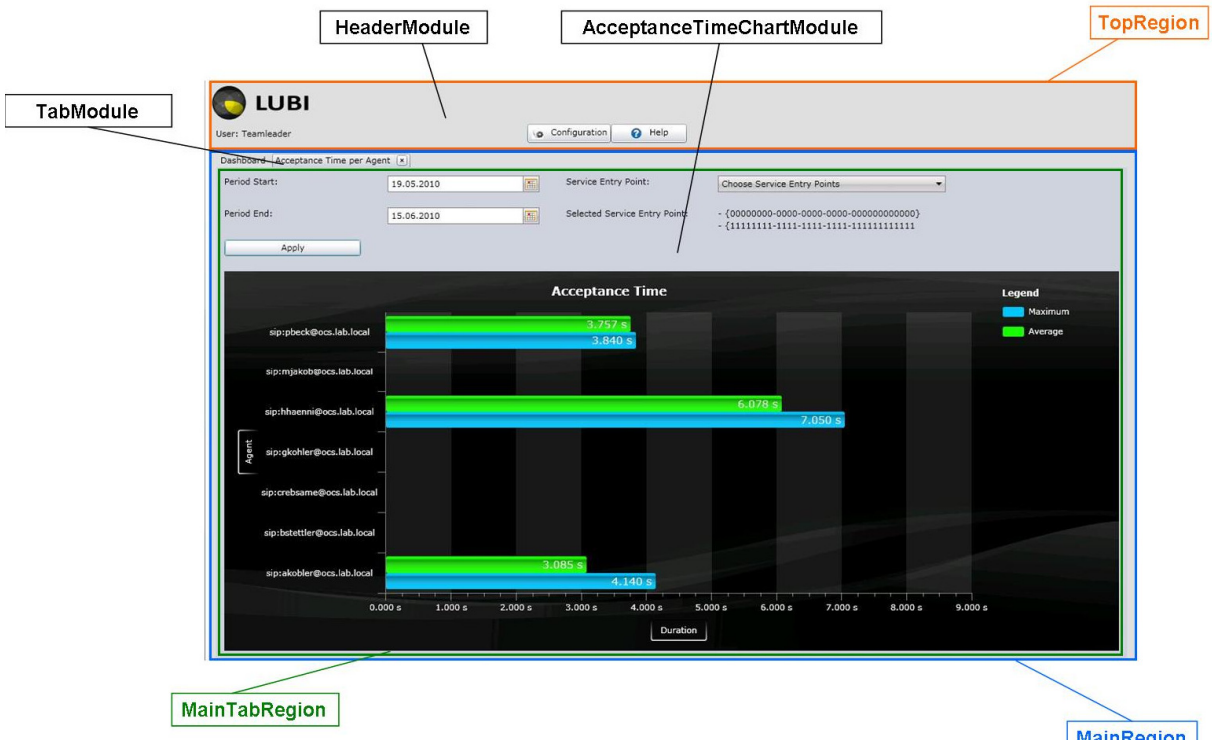


Abbildung 92: Prism Regions und Modules

Im XAML-Code werden Regions folgendermassen definiert:

```
<ContentControl x:Name="TopRegion"
                Regions:RegionManager.RegionName="TopRegion"
                Grid.Row="0"
                HorizontalAlignment="Left"
                MinHeight="50"
                Height="80"
                VerticalAlignment="Top" />
```

Für das Laden eines Moduls in eine Region ist folgender Code nötig:

```
unityContainer.RegisterType<IDashboardViewModel, DashboardViewModel>();
regionManager.RegisterViewWithRegion("MainTabRegion", typeof(DashboardViewModel));
```

Mit der RegisterType()-Methode wird dem Container der Typ des ViewModels bekannt gemacht, was dem Container erlaubt, zu gegebener Zeit eine Instanz des ViewModels zu erzeugen, sie per Dependancy Injection oder per Resolve()-Methode in den Container zu laden. Mit der Methode RegisterViewWithRegion wird das ViewModel, welches wiederum die View erstellt, im RegionManager in der spezifizierten Region registriert.

11.2.2 Telerik-Controls[30]

Die Firma Telerik verkauft mehrere Bibliotheken mit Silverlight-Controls. Das INS hat für ein früheres Projekt diese Bibliotheken lizenziert und für diese Bachelor-Arbeit zur Verfügung gestellt. Einige Controls wurden für die Umsetzung des UI's benutzt.

11.2.2.1 RadChart

Das mit Abstand am meisten verwendete Control ist das RadChart. Es wird für jegliche Chart-Darstellung im LUBIWebFrontend verwendet. Die Konfigurations- und Gestaltungsmöglichkeiten des RadChart sind mannigfaltig. Für eine umfangreiche Dokumentation wird auf die Website[31] und das Forum von Telerik verwiesen[32]. Einige Punkte sollen aber hervorgehoben werden:

[30] Weitere Informationen: http://www.telerik.com/products/silverlight.aspx
[31] Weitere Informationen: http://www.telerik.com/help/silverlight/radchart-overview.html
[32] Weitere Informationen: http://www.telerik.com/community/forums.aspx

Wird die DefaultView des Charts deklarativ, sprich in XAML verändert, muss anschliessend die Legende ebenfalls deklariert werden. Sie wird nicht mehr automatisch erstellt.

Code-Beispiel:

```
<telerikcharting1:RadChart.DefaultView>
  <telerikcharting2:ChartDefaultView>
     <telerikcharting2:ChartDefaultView.ChartTitle>
        <telerikcharting2:ChartTitle Content="{Binding ChartTitle}"/>
     </telerikcharting2:ChartDefaultView.ChartTitle>
     <telerikcharting2:ChartDefaultView.ChartLegend>
       <telerikcharting2:ChartLegend Name="chartLegend" UseAutoGeneratedItems="True"
        Header="Legend"/>
     </telerikcharting2:ChartDefaultView.ChartLegend>
     <telerikcharting2:ChartDefaultView.ChartArea>
        <telerikcharting2:ChartArea LegendName="chartLegend">
     </telerikcharting2:ChartArea>
     </telerikcharting2:ChartDefaultView.ChartArea>
  </telerikcharting2:ChartDefaultView>
</telerikcharting1:RadChart.DefaultView>
```

Das RadChart-Control bietet verschiedene Möglichkeiten, Daten als ChartSeries zu mappen. Für die Charts des LUBIWebFrontend wurde das SeriesMapping manuell erstellt, um möglichst grosse Kontrolle über die angezeigten Werte und Bezeichnungen zu erlangen.

Code-Beispiel:

```
<telerikcharting1:RadChart.SeriesMappings>
  <telerikcharting2:SeriesMapping>
    <telerikcharting2:SeriesMapping.SeriesDefinition>
      <telerikcharting2:HorizontalBarSeriesDefinition ShowItemToolTips="True"
    ShowItemLabels="True" ItemLabelFormat="#\.000 s"/>
    </telerikcharting2:SeriesMapping.SeriesDefinition>
    <telerikcharting2:SeriesMapping.GroupingSettings>
      <telerikcharting2:GroupingSettings>
        <telerikcharting2:GroupingSettings.GroupDescriptors>
          <telerikcharting2:ChartGroupDescriptor Member="Category"/>
        </telerikcharting2:GroupingSettings.GroupDescriptors>
      </telerikcharting2:GroupingSettings>
    </telerikcharting2:SeriesMapping.GroupingSettings>
    <telerikcharting2:SeriesMapping.ItemMappings>
      <telerikcharting2:ItemMapping DataPointMember="YValue" FieldName="Value"/>
      <telerikcharting2:ItemMapping DataPointMember="XCategory" FieldName="Name"/>
    </telerikcharting2:SeriesMapping.ItemMappings>
  </telerikcharting2:SeriesMapping>
</telerikcharting1:RadChart.SeriesMappings>
```

Im ItemMapping wird angegeben, als was der Wert, welcher im Field mit dem gesetzten Namen steht, im Chart abgebildet wird. Es gibt die Möglichkeit, den Wert auf verschieden Attribute zu binden. Im LUBIWebFrontend wurde auf folgende Attribute gebunden:

- XValue: Bindet zum X-Wert des Datenpunkts
- YValue: Bindet zum Y-Wert des Datenpunkts
- XCategory: Zur Katgorie des Datenpunkts auf der X-Achse
- LegendLabel: Bindet zum Legendeneintrag des Datenpunkts

Das RadChart unterstützt als Werte auf den Achsen grundsätzlich nur Zahlen-Werte. Es gibt jedoch die Möglichkeit, auf der X-Achse DateTime-Werte zu verwenden, die vom Chart für die Darstellung korrekt umgerechnet werden. Wünschenswert wäre diese Unterstützung auch auf der Y-Achse, um Charts mit Zeitdauern als Werte im korrekten Format (z.B. mm:ss) darstellen zu können. Noch schöner wäre die Unterstüzung des TimeSpan-Datentyps. Da diese Möglichkeit wie erwähnt zurzeit nicht besteht, wird eine Zeitdauer momentan in Minuten und Sekunden dezimal angegeben, sprich 5.50 min entspricht 5 min 30 sec.

11.2.2.2 RadTabControl

Das RadTabControl ist ein Navigations-Control, welches die Benutzung von Tabbed Interfaces in Silverlight ermöglicht. Damit ist ein flexibler Umgang mit verschiedenen Sichten der Applikation möglich. Im LUBIWebFrontend wird das Dashboard in einem nicht schliessbaren Tab angezeigt, während die Detail-Ansichten der Charts geschlossen werden können. Dies dient einer grösseren Übersichtlichkeit.

Um das RadTabControl als Region-Host in Prism verwenden zu können, muss ein RegionAdapter geschrieben werden. Im LUBIWebFrontend konnte dabei auf die Vorarbeit von Michael Jakob zurückgegriffen werden, der einen solchen Adapter bereits in einem anderen Projekt erstellte.

Der RegionAdapter erbt von *RegionAdapterBase<T>* wobei T der Datentyp des Controls ist, welches als Region-Host verwendet werden will.

Die Methoden CreateRegion() und Adapt() müssen anschliessend überschrieben werden.

```
protected override void Adapt(IRegion region, RadTabControl regionTarget)
{
  bool itemsSourceIsSet = regionTarget.ItemsSource != null;

  if (itemsSourceIsSet)
  {
    throw new InvalidOperationException("ItemsControlHasItemsSourceException");
  }
}

protected override IRegion CreateRegion()
{
  return new SingleActiveRegion();
}
```

Die Methode CreateRegion legt fest, wie sich die Region verhalten soll. Folgende Möglichkeiten bestehen:

- Region: Mehrere aktive Views in der Region erlaubt. Für Controls, welche von Selector abgeleitet sind
- SingleActiveRegion: Nur eine aktive Region erlaubt. Für Controls, welche von ContentControl ableiten
- AllActiveRegion: Alle Views sind immer aktiv, Deaktivierung ist nicht erlaubt. Für auf ItemsControl-basierenden Controls.

Um den RegionAdapter zu verwenden, wird im Bootstrapper die Methode ConfigureRegionAdapterMappings() überschrieben:

```
protected override RegionAdapterMappings ConfigureRegionAdapterMappings()
{
  RegionAdapterMappings regionAdapterMappings = base.ConfigureRegionAdapterMappings();

  if (regionAdapterMappings != null)
  {
    regionAdapterMappings.RegisterMapping(typeof(RadTabControl),
  Container.Resolve<RadTabControlRegionAdapter>());
  }
  return regionAdapterMappings;
}
```

12 Ausblick

Dieses Kapitel beschreibt Punkte, die in dieser Arbeit aus zeitlichen Gründen nicht mehr realisiert werden konnten. Die genannten Punkte werden der Vollständigkeit halber genannt und sind als Vorschläge für weitere Entwicklungen in LUBI zu verstehen.

12.1 Bereich LUBI

12.1.1 Bestehende Dimensionen ausbauen

Die Granularität der zurzeit bestehenden Dimensionen könnte teilweise noch vergrössert werden. Dadurch ergeben sich weitere Aspekte. Beispielsweise wären in der Time-Dimension Angaben über die Woche des Jahres, den Tag der Woche oder Ähnliches denkbar. Damit wird man flexibler in den Möglichkeiten der Abfrage. Man könnte so zum Beispiel die Frage nach der Conversation-Verteilung verschiedener Montage beantworten.

12.1.2 Weitere Dimensionen erstellen

Die Erstellung weiterer Dimensionen wäre interessant, um die Geschäftsdaten unter weiteren Aspekten abfragen zu können. So wäre z.B. eine Language-Dimension interessant, mit der die Konversationen in verwendete Sprachen klassifiziert werden können. Oder eine Status Dimension, die auf die Zahlen der Kundenberater angewendet wird. Damit liessen sich Berichte über den Status der Kundenberater erstellen. Voraussetzung für alle Erweiterungen der Dimensionen eines Cubes ist selbstverständlich das Vorhandensein der entsprechenden Daten in der Live-Datenbank.

12.1.3 Weitere Management-Reports

Neben den bereits erstellten Standart-Management-Reports könnten weitere Reports erstellt werden um zusätzliche Aspekte eines Contact-Centers abzudecken.

12.2 Bereich LUBIWebFrontend

12.2.1 User Experience verbessern

Die User Experience des LUBIWebFrontend-UI soll verbessert werden. Die Wartezeit, bis die Daten vom Webservice an die Silverlight-Applikation übertragen wurden, soll für den User mit einer Statusbar angezeigt werden. Das gesamte Styling der Applikation soll überarbeitet werden und ans Styling der Luware-Lösung angepasst werden. Die Ausnutzung hoher Bildschirmauflösungen soll ebenfalls implementiert werden.

12.2.2 Dashboard überarbeiten

Das Dashboard, welches sich zurzeit nicht sehr übersichtlich präsentiert, soll verbessert werden. Es muss eine Lösung für die Grösse der Charts gefunden werden. Momentan lassen sich die Charts nicht kleiner darstellen, ohne das essentielle Teile oder Daten unleserlich werden. Einen Ansatz bietet hier der neue Release der Telerik-Controls, welche das Arrangement der Legende auch unterhalb des Charts erlauben. Ob weitere Verbesserungen mit dem Styling der Charts erreicht werden können, ist abzuklären.

Ein weiterer Punkt ist die Implementierung einer Drag & Drop-Reorder-Möglichkeit für die Charts. Dies würde jedem Benutzer erlauben, sein eigenes Dashboard zu konfigurieren. Dazu gehört auch die Möglichkeit, für den Benutzer uninteressante Charts auszublenden und gegebenenfalls auf Wunsch wieder einzublenden.

12.2.3 Einbettung in Authetifizierungslösung

Um die Dashboards und auch die angezeigten Daten zu personalisieren, muss die LUBIWebFrontend-Lösung in eine Authentifizierungslösung eingebettet werden. Anschliessend kann im LUBIWebFrontend die Authorisierung für Charts und Daten durchgeführt werden.

12.2.4 Neue Charts

Werden die Dimensionen des Cubes angepasst oder erweitert, können zusätzliche Charts für das LUBIWebFrontend erstellt werden. Denkbar ist z.B. ein Chart, welches darstellt, welcher Kundenberater des Teams wieviele Konversationen in welcher Sprache geführt hat.

12.2.5 Live-Wall

Für das Real-Time-Reporting, oder wenigstens das Near-Real-Time-Reporting, ist eine Live-Wall zu implementieren. Der Datenzugriff erfolgt hier nicht auf die Analysis Services, da die Daten dort nur tagesaktuell sind, sondern direkt auf die Datenbank. Die Integration der Live-Wall in das LUBIWebFrontend ist vertieft zu prüfen. Für diese Integration ausgelegt wäre das LUBIWebFrontend durch die Modularisierung mit Prism.

13 Glossar

13.1 allgemein

ACD	Automatic Call Distribution
AS	Analysis Services
BI	Business Intelligence
BIDS	Business Intelligence Development Studio
CRM	Customer Relationship Management
DMX	Data Mining Extension
DSV	Data Source View
ETL	Extract, Transform, Load
ERP	Enterprise Resource Planning
ICM	Intelligent Contact Management
IS	Integration Services
KPI	Key Performance Indicator
MDX	Multidimensional Expression
OCS	Office Communications Server
OLAP	Online Analytical Processing
OLTP	Online Transactional Processing
SSAS	SQL Server Analysis Services
SSIS	SQL Server Integration Services
SSRS	SQL Server Reporting Services
SCM	Supply Chain-Management
UDM	Unified Dimensional Model

13.2 projektspezifisch

LUCSM	Lean Unified Customer Service Manager
AM	Activity Monitor
ICH	Interactive Conversation Handler
ICR	Interactive Conversation Response
LUCC	Lean Unified Conversation Context
LUAC	Lean Unified Agent Controller
LUBI	Lean Unified Business Intelligence
LURW	Lean Unified Reporting Wall
LUWF	Lean Unified Workflow
PBCD	Policy Based Call Distribution

14 Literaturverzeichnis

[BI] Resort. (15. 06 2010). Abgerufen am 15. 06 2010 von Blog "BI SSAS - What is an OLAP cube?": http://biresort.net/blogs/pedrocgd/archive/2009/06/04/bi-ssas-what-is-a-cube.aspx

Brosius, Azevedo, Dehnert, Neumann, & Scheerer. (2009). BI & Reporting, S. 47. In P. A. Gerhard Brosius, *Business Intelligence und Reporting mit SQL Server 2008* (S. 47).

Gorbach, Melomed, Berger, & Bateman. (2006). Microsoft SQL Server 2005 Analysis Services. In Gorbach, Melomed, Berger, & Bateman, *Microsoft SQL Server 2005 Analysis Services* (S. 586-600). USA: Sam's Publishing.

Microsoft. (09. Juni 2010). *Abrufen von Daten einer analytischen Datenquelle*. Abgerufen am 09. Juni 2010 von Abrufen von Daten einer analytischen Datenquelle: http://msdn.microsoft.com/de-de/library/ms123479%28v=SQL.105%29.aspx

Microsoft. (09. Juni 2010). *Clients (Analysis Services - Mehrdimensionale Daten)*. Abgerufen am 09. Juni 2010 von Clients (Analysis Services - Mehrdimensionale Daten): http://msdn.microsoft.com/de-de/library/ms174518.aspx

Microsoft. (25. 05 2010). *Technical Concepts*. Abgerufen am 25. 05 2010 von Technical Concepts: http://msdn.microsoft.com/en-us/library/ff649897%28v=pandp.10%29.aspx

Ross, M., & Kimball, R. (2002). *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling.*

Wikipedia - Snowflake Schema. (24. 03 2010). Abgerufen am 24. 03 2010 von Wikipedia - Snowflake Schema: http://en.wikipedia.org/wiki/Snowflake_schema

Wikipedia - Star Schema. (24. 03 2010). Abgerufen am 24. 03 2010 von Wikipedia - Star Schema: http://en.wikipedia.org/wiki/Star_schema

15 Abbildungsverzeichnis

16 Tabellenverzeichnis