Studiengang: Master of Science an der HAW-Hamburg
Vorlesung: Projekt „Ferienclub“
Ausarbeitung von: Thomas Steinberg

HAW HAMBURG

Zusammenfassung des Projekts „Ferienclub“ der Teilgruppe Transformation Engine

Thomas Steinberg, 15. Februar 2006
thomas.steinberg@informatik.haw-hamburg.de

Die dazugehörige Präsentation wurde im Projekt des Studiengangs Master of Science von Thomas Steinberg am 02.02.2006 gehalten. Diese Ausarbeitung besteht aus einer allgemeinen Einführung in das Thema, der kurzen Vorstellung der Ziele aus Anwendung 1 und der überarbeiteten Ziele direkt im Projekt in Voraussicht auf die Master Thesis. Hinzu kommen prinzipielle Einleitungen in die Fachgebiete und die einzelnen Implementierung anhand von drei Versionen um die Fortschritte zu dokumentieren. Am Ende der Ausarbeitung folgt ein Ausblick auf die Master Thesis, wo die Grundlagenforschung des Projekts die ersten Erkenntnisse bringen sollte.

Kurzzusammenfassung

Dieses Projekt umfasst zwei große Themen, zu einem wäre da die Transformation von XML-Dokumenten und zum anderen Web Services.

Die meisten strukturierten Daten werden mit Hilfe von XML-Dokumenten übertragen. Da diese aber heute noch kaum von einem Browser unterstützt werden oder ordentlich in einem lesbaren Format bearbeitet werden können, müssen diese erst transformiert werden. Die physische Auszeichnung der XML-Dokumente kann durch die Transformation über XSL [i]erfolgen. Mit XSL ist es möglich, Daten im XML-Format über verschiedene XSL-Transformationen in andere Dokumentenformate zu überführen. Sehr häufig wird die Transformation in das HTML-Format durchgeführt. Es existieren verschiedene Möglichkeiten, um z.B. eine Transformation von XML-Daten in die Formate SVG oder PDF durchzuführen.

Die Idee hinter Web Services ist auch nicht neu. Übertragungsstandards sorgen dafür, dass Daten und Funktionalität zwischen Kommunikationspartnern ausgetauscht werden können. Die Kommunikationspartner dabei sind keine Menschen, sondern alle Maschinen. Daraus leitet sich der Name Maschine-zu-Maschine-Kommunikation ab. Eine Maschine ist in diesem Fall eine Anwendung, die beispielsweise auf einem Webserver liegt.

Einleitung

Als erstes sollte der Ablauf des Studiengangs für externe vorgestellt werden. Die Idee ist, dass man in der Vorlesung „Anwendungen 1“ im zweiten Semester sich ein Thema für die Master Thesis ausdenkt. Nach der erfolgreichen Findung, soll man sich in das Thema theoretisch einarbeiten und die Ziele grob einteilen. Im „Projekt“ des dritten Semesters werden einige theoretischen Ziele auf Basis von praktischer Grundlagenforschung schon mal behandelt und durch einen Prototypen realisiert. Letztendlich soll man sich durch das gesamte Studium schon mit einem Thema außer einander setzen und so viel wie möglich an Vorwissen sammeln, um die Zeit für die Master Thesis optimal zu nutzen und so viele Einbahnstrassen wie möglich im Vorwege auszuschalten.

Aus dem Grund das sich mein Master Thesis Thema in der Zwischenzeit des Projekts geändert hat, haben sich auch meine Schwerpunkte während des Projekts geändert. Somit kommt es zu der Einteilung der Kapitel in „Ziele Anwendung 1“ und „Ziele neu definiert“. Erwähnt werden sollte auch, dass das Szenario hier eine Mietung eines Autos von einer Autovermietung nur temporal besteht um sich im Projekt Ferienclub zu integrieren und nicht in der Master Thesis weiter verfolgt wird.

Szenario

Wir befinden uns in einem smarten Ferienclub, wo jeder Feriengast über einen persönlichen PDA[ii] mit der Ferienclubzentrale verbunden ist. Auf diesen PDA befindet sich ein Terminkalender, in den der Gast all seine Aktivitäten für die zwei Wochen die er im Club ist eintragen kann. Stellen wir uns vor der Gast möchte am zweiten Tag seines Urlaubs sich die Umgebung anschauen und benötigt ein Auto. Für diesen Zweck gibt es eine Autovermietungsmaske erstellt vom Ferienclub, in die er eintragen kann, welche Eigenschaften das Auto haben soll.
Was steht hinter diesem Angebot des Ferienclubs. Die Suchmaske wurde von einem Entwickler des Ferienclubs entwickelt. Die gewonnen Daten werden in einer Datenbank des Ferienclubs nach einen proprietäre Schema abgespeichert. Jetzt will der Ferienclub den Gast zufrieden stellen und braucht bestimmte Daten von Autovermietungen aus der Umgebung. Im Allgemeinen sitzt nicht der gleiche Entwickler an den Softwaresystem der Autovermietung und somit stimmen die Datenstrukturen nicht überein. Was muss geschehen, damit der Ferienclub mit der Autovermietung kommunizieren kann? Der Autovermieter muss sein Angebot auf irgendeine Weise zu Verfügung stellen und publik machen. Der Ferienclub braucht ein System, das seine Datenstrukturen transformiert in die jeweiligen Datenstrukturen, die von dem Autovermieter publik gemacht werden.

Hier setzt das Projekt an. Es wird eine Transformation Engine erzeugt, die die Datenstruktur auf syntaktischer Ebene in die Datenstruktur des Autovermieters transformiert. Natürlich soll dies bidirektional funktionieren. Des Weiteren wird der Service, der Transformation Engine und des Autovermieters publik gemacht, indem die Services als Web Services implementiert werden sollen.

Grundlagen

Transformation

Die Bearbeitung und Transformation von XML-Dokumenten benötigt einige Standards. Als erstes sollte der ***Parser***[iii] erwähnt werden. Dieser stellt die Gültigkeit sowie die Wohlgeformtheit eines XML Dokuments fest. Hierzu wird die DTD[iv] benötigt um die Strukturen der Elemente abzugleichen. Nach dem Test des Parser, liefert dieser eine Datenstruktur mit der eine entsprechende Anwendung arbeiten kann. Im Wesentlichen gibt es zwei wichtige Unterscheidungsformen bei Parsern. Zum einen gibt es die Klasse der SAX[v] Parser (Simple API for XML) und zum anderen die Klasse der DOM[vi] Parser (Document Object Model) hier *Xerces* [vii]*2*. Welchen Parser man nutzt ist einzig und allein von der Anwendung abhängig. Im Prinzip arbeitet SAX ein XML Dokument immer sequentiell ab und kennt nur die Elemente an der aktuellen Stelle im XML Baum. DOM hingegen hält zur Laufzeit den kompletten XML-Baum des Dokuments im Speicher.

Um XML-Dokumente zu transformieren braucht man einen weiteren Standard ***XSL*** *(Extensible Stylesheet Language)*. XSL basiert syntaktisch gesehen auf XML und ermöglicht die Generierung von beliebigen Dokumenten aus einer Basis von strukturierten XML Daten. Dabei werden die eigentlichen Formatierungsanweisungen als XSL und der Mechanismus der Überführung in ein neues Format als ***XSLT***[viii] *(Extensible Stylesheet Language Transformation)* bezeichnet.“ Die eigentliche Transformation wird mit Hilfe des so genannten Stylesheet [ix]durchgeführt. Dieses Stylesheet legt fest, wie die Daten umzuwandeln sind. Die Umwandlung selbst führt ein XSLT Prozessor durch, welcher aus Stylesheet und XML Daten ein entsprechend neues Format generiert. Die Apache Group stellt hier einen XSLT Prozessor namens „*Xalan*[x]“ zu Verfügung.
Der Ablauf der Transformation ist wie folgt, der XSLT Prozessor arbeitet Knoten für Knoten des Quelldokument ab. Zur Navigation bzw. Adressierung der Daten verwendet XSLT gültige XPath [xi]Ausdrücke und Funktionen. Mit Hilfe des Stylesheet wird nun anhand einer Schablone (<xsl:template…>) festgelegt was passieren soll, wenn ein Knoten gefunden wird.

Um eine komplexe Transformation z.B. in ein PDF-Format braucht man ein ***FO***[xii] *(Formating Objects)* als eine Erweiterung der XSL Sprache, die es ermöglicht komplexe Formatierungen durchzuführen. Mit Formating Objects wird genau beschrieben, wie das Dokument auszusehen hat und wie die Inhalte der jeweiligen XML Knoten formatiert werden sollen. „Verwendet ein XSL Stylesheet die FO Erweiterung, spricht man auch von einem XSL-FO Stylesheet. Steht ein solches XSL-FO Stylesheet zur Verfügung, ist es nun möglich eine XML Datenbasis mittels einer XSLT Transformation durchzuführen. Das Ergebnis ist dann eine XML Ausgabe, welche die FO Erweiterung nutzt. Die FO Erweiterung beschreibt nun exakt, wie das Ergebnisdokument auszusehen hat. Anhand dieses FO Dokuments lässt sich dann ein beliebiges Format wie z.B. PDF erzeugen.“

Web Service

Das Thema Web Services ist natürlich komplexer, als dass man es hier auf einer Seite darstellen kann. Dies soll nur eine kurze Einführung in das Thema Web Services mit AXIS’s werden.

Allgemein ist ein Web Service eine Software Anwendung die auf einem Server zu Verfügung gestellt wird. Diese Anwendung ist über eine Uniform Resource Identifier (URI) eindeutig identifiziert. Ein Anbieter der diese Anwendung (seinen Service), anderen Benutzern zu Verfügungen stellen möchte, muss diesen in einem Verzeichnis veröffentlichen. ***UDDI***[xiii] *(Universal Description, Discovery and Integration)* wird als ein Verzeichnisdienst zur Registrierung von Web Services verwendet. Es ermöglicht das dynamische Finden des Web Services durch den Konsumenten. Mit dem veröffentlichen ist es noch nicht getan, der Dienst muss noch beschrieben werden, damit dem Konsumenten die unterstützenden Methoden und deren Parameter bekannt gegeben werden. Dies geschieht über die ***WSDL***[xiv] *(Web Services Description Language)*. Jetzt kann auf den Web Service von Seiten des Konsumenten unabhängig von seiner Programmiersprache zugegriffen werden. Hierfür wird ***SOA*** *(Service Oriented Architecture)* oder ***XML-RPC***[xv] *(XML-Remote Procedure Call)* verwendet.
Web Services sind eigentlich nicht für menschliche Benutzer gedacht auch wenn sie (menschen-)lesbar implementiert werden, sind sie hauptsächlich für Softwaresysteme, die automatisierte Daten austauschen und/oder Funktionen auf entfernten Rechnern aufrufen.

Nach der groben Einführung in Web Services, kommt hier noch eine kurze Erläuterung zum Open Source Projekt ***AXIS***[xvi] *(Apache eXtensible Interaction System)*. AXIS ist ein Nachfolgeprojekt der Apache SOAP Implementierung. Zu Vergleich zu SOAP, das auf dem langsamen DOM basiert, verwendet AXIS SAX Filterketten. Im Punkto Funktionalität, Performanz und Interoperalität hat AXIS, Apache SOAP Implementierung bereits überrundet. Ein weiterer Vorteil ist die Ausrichtung an die JAX-RPC Spezifikation von Sun.
Die Architektur ist folgendermaßen aufgebaut. „Module gliedern AXIS in Subsysteme, die verschiedene Aufgaben übernehmen. Die Verarbeitung von Anfragen erfolgt über Filterketten, die sich aus Handlern zusammensetzen. Bei der Programmentwicklung ist es nicht notwendig selbst in die Filterketten einzugreifen. Mit dem Konzept der Ketten ist AXIS jedoch offen für Erweiterungen.

Damit Web Services bereitgestellt werden können, wird ein Server benötigt, der Web Anfragen verarbeiten kann. AXIS kann in einen Web Container integriert werden. Es gibt dafür eine Web Anwendung, die AXIS in Form von drei Servlets beinhaltet. Das AXIS Servlet kann Web Services aufnehmen und Anfragen an diese zur Weiterverarbeitung routen. Aufgrund der Zustellungsfunktion spricht man von einem SOAP Router oder Dispatcher. Das Admin Servlet dient der Fernwartung des SOAP Routers. Web Services können über dieses Servlet installiert und deinstalliert werden.“

Ziele

Ziele Anwendung 1

Da die Ziele aus Anwendung 1 zum Projekt dazugehören, werden sie in diesem Kapitel kurz vorgestellt.

Das „BIG Picture“ aus Anwendung 1 war, eine „Transformation Engine[xvii]“ zu bauen um die einzelnen Aktoren im Ferienclub, die unterschiedliche Protokolle fahren und verschiedenen Schnittstellen haben über eine „Transformation Engine“ auf Basis von XML-Transformation durch XSLT mit dem Ferienclubmanagement zu verbinden.

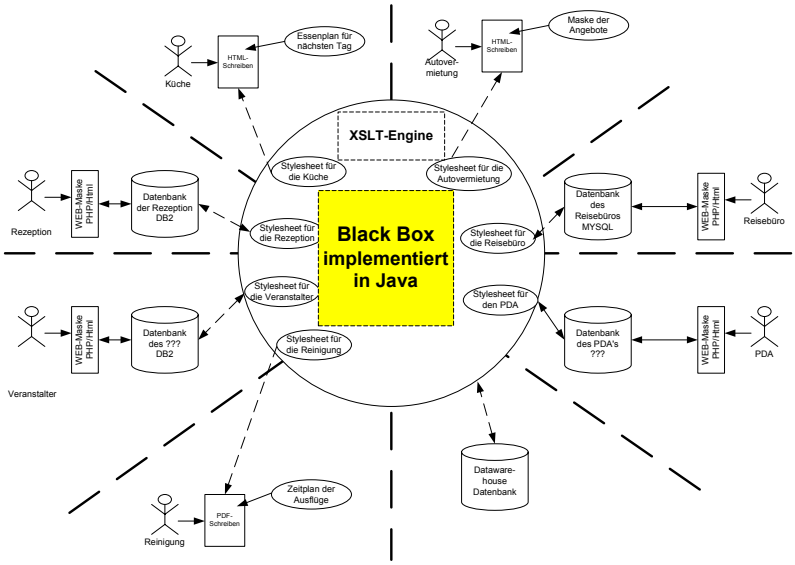


Abb. 1: Big Picture aus Anwendungen 1

Die erste Implementation (**Version 1.0**) der „Transformation Engine“, sollte mit Hilfe einer Zuordnungsmaske (am Beispiel von Outlook-Synchronisation) die ersten Erfahrungen im Projekt erbringen. Es sollte eine Oberfläche (siehe Abb. 2: Big Picture Version 1) in Java gebaut werden, in die vom User-Hand die richtige Zuordnung der einzelnen Elemente zweier XML Dokumente (bzw. einen anderen Destination Dokument) darstellt. Mit der Zuordnungstabelle werden in der ersten Implementation dem Problem *Synonyme* und/oder *Homonyme* aus dem Weg gegangen. Bei betätigen des “Create Buttons“ und der Auswahl des Ziel Dokuments wird eine Transformation ausgeführt und ein neues Dokument in dem entsprechenden gewünschten Format erzeugt.

Als eine Erweiterung der ersten Implementierung (**Version 1.1**) war ein “loadIn“ vorgesehen. Was ist ein loadIn? LoadIn sollte ein Feature sein, um bei gleichen Zuordnungstabellen dem User den Aufwand abzunehmen und eine entsprechende Zuordnungstabelle(die z.B. in einer Textdatei gespeichert ist) in die „Transformation Engine“ Reinzuladen.

Ein zusätzliches Angebot an das Datawarehouse / Dataminig Team, sollte die „Transformation Engine“ in der (**Version 2.0**) als Modul eine Visualisierung der im XML-Dokument enthaltenen Daten(Knoten) als Torten/- Kuchendiagramme oder sonstige Diagrammtypen implementiert bekommen. Mit Hilfe von Batik und SVG wird der alternative Weg eingeschlagen, hier sollten wiederum neue Erkenntnisse auf der Visualisierung von Daten / Informationen im Projekt gesammelt werden.

Die Implementation (**Version 3.0**) der „Transformation Engine“ sollte in Kooperation mit Artem Khvat stattfinden, sie sollte mit Hilfe von Semantik durch Ontologien automatisch eine Transformation tätigen. Dies setzt zu mindestens eine vollständig implementierte Grundstruktur durch die Version 1.0 voraus. Es sollte untersucht werden, in wie weit man Daten semantisch kennzeichnen kann, sodass man ohne fremde/zusätzliche Hilfe, Dokumente auf syntaktischer Ebene transformieren kann.

Ziele neu definiert

Wie in der Einleitung schon erwähnt hat sich das Thema für die Master Thesis komplett geändert, wobei der Schwerpunkt der Master Thesis auf die Kommunikation ausgerichtet wurde. Da die Master Thesis nicht in zwei Sätzen zu erklären ist, wird ein Verweis auf die Ausarbeitung zum Thema „Master Thesis Outline“ aus der Seminarvorlesung gegeben. Es mussten neben der neuen theoretischen Grundlagenforschung für das neue Master Thema auch die Ziele während des Projekts dementsprechend umformuliert oder sogar neu definiert werden. Somit haben sich einige Grundlegende Ziele geändert.

Die ***Version 1.0*** ist wie im vorherigen Kapitel beschrieben gleich geblieben, um den Umgang mit den Grundfunktionen der Transformation zu erlangen. Allerdings wurde die ***Version 1.1*** wegen Zeitgründen fallen gelassen.

Die ***Version 2.0*** wurde komplett gestrichen in Absprache mit dem BI-Team [xviii]und komplett umgestaltet. Die Idee zu Version 2 entstand während der Überlegungen zu Version 3, wo man auf eine Art mit der „semantischen Engine[xix]“ von Artem Khvat kommunizieren musste. Außerdem war die Idee, die „Transformation Engine“ als einen Web Service der SOAP-Gruppe [xx] zu Verfügung zu stellen, während des Projekt entstanden. Es war notwendig sich in das Thema Web Service kurzfristig einzuarbeiten, um folgende Aufgaben erfüllen zu können.
Es sollte möglich sein, aus der SOAP-Architektur [xxi] heraus eine Anfrage auf ein Auto an eine beliebige Autovermietung, mit der Datenstruktur (XML-Dokument) des Ferienclubs, zu schicken. Die „Transformation Engine“ sollte die Transformation der Datenstruktur übernehmen und das entsprechend neue XML-Dokument an eine bestimmte Autovermietung weiterleiten. Den Weg der Response-Nachricht zurück von der Autovermietung zum Ferienclub sollte synchron ablaufen, indem die „Transformation Engine“ die Datenstruktur des XML-Dokuments in die Datenstruktur des Ferienclubs wieder ändert. Da die SOAP-Architektur mit Web Service arbeitet und Web Service eine grundlegende Funktion in der Master Thesis spielen könnte, wurde beschlossen die „Transformation Engine“ und die drei unterschiedlichen Dummy-Autovermietungen als Web Service zu implementieren.

Die ***Version 3.0*** ist wie im vorherigen Kapitel beschrieben auch gleich geblieben, um den Umgang mit den Grundfunktionen der semantischen Transformation zu erlangen.

Prototypen

Entwicklungsumgebung

In diesem Kapitel wird beschrieben was man benötigt um die Prototypen Version 1.0 bis 3.0 zu realisieren. Hierzu wurden benutzt:

Java Entwicklungsumgebung: Eclipse 3.1 [13.]
Java Api’s [12.]:

- *JAXM* [12.] (Einfaches Empfangen und versenden von XML-basierten Nachrichten, basierend auf SOAP) „angesehen“.

- *JAXP 1.2* [12.] (Unterstützt die Verarbeitung von XML-Dokumenten unter Nutzung von SAX, DOM und XSLT) „aktiv benutzt“.
  → TrAX-API benötigt Xerces 2 (DOM-Parser) → Enthält Xalan (XSLT-Prozessor)

- *JAX-RPC* [12.] Java API for XML-based RPC (Stellt das Basis-Api zur Entwicklung und zur Weitergabe von Web Service zur Verfügung) „angesehen“.

- *SAAJ SOAP* [12.] with Attachments API for Java (Erzeugen und Empfangen von Nachrichten, die auf der SOAP-1.1-Spezifikation basieren, Unterstützung von SOAP mit Anhängen) „angesehen“.

Benutzt man die J2SE 1.4.1 (die JAXP1.1 enthält) die bereits ein Parser im Package *javax.xml* implementiert, muss man die neuste Version 1.2 von JAXP separat herunterladen und den Xerces-Parser separat installieren. Laut einer Ankündigung wird in der neuen Version der J2SE 1.5 der Xerces 2 [11.] (JAXP 1.2) schon verwendet.

Um Web Services bereitzustellen wird ein Server benötigt, der Anfragen bearbeiten kann. Um umfangreiche Tools zur Web Service Erzeugung und Bearbeitung als Hilfsmittel zu haben, eignet sich die Architektur von AXIS [14.] hervorhangend. AXIS [14.] wird in einen Web Container, hier Tomcat 5 [9.] integriert. Es gibt dafür eine Web Anwendung, die AXIS in Form von drei Servlets beinhaltet.

Die schon erwähnten Hilfswerkzeuge sind für die Erzeugung von Web Services sehr nützlich. Um eine Generierung von Proxys aus WSDL zu erzeugen. Benötigt man eine WSDL, diese WSDL Dateien können für installierte Web Services abgefragt werden. Die WSDL Beschreibung des Web Services wird automatisiert erzeugt. Zum Abfragen reicht ein einfacher Web Browser.
Auch in die andere Richtung vom vorhandenen Client in Java zu einem Web Service gibt es Hilfsmittel, denn WSDL Dateien sind recht kompliziert aufgebaut und eignen sich nicht für eine Erstellung von Hand. Das AXIS Werkzeug Java2WSDL kann aus normalen Java Klassen eine WSDL Datei generieren. Die WSDL Datei kann über einen Web Server veröffentlicht werden, um Fremden die Nutzung eines Web Services zu erleichtern.

Als nächstes wird aus einer WSDL eine Client bzw. eine Server Klasse erzeugt, hierfür gibt es auch Hilfsmittel in AXIS [14.]. Das Werkzeug WSDL2Java erzeugt aus einer WSDL Beschreibung, die Java-Klassen für den Client bzw. den Server. Für den Server werden Rumpfklassen erzeugt, mit deren Hilfe sich leicht Serverfunktionalität ansprechen lässt. Mit den Client Klassen kann mit wenigen Zeilen Code ein Client für einen bestehenden Web Service erstellt werden.

Prototypenaufbau

Version 1

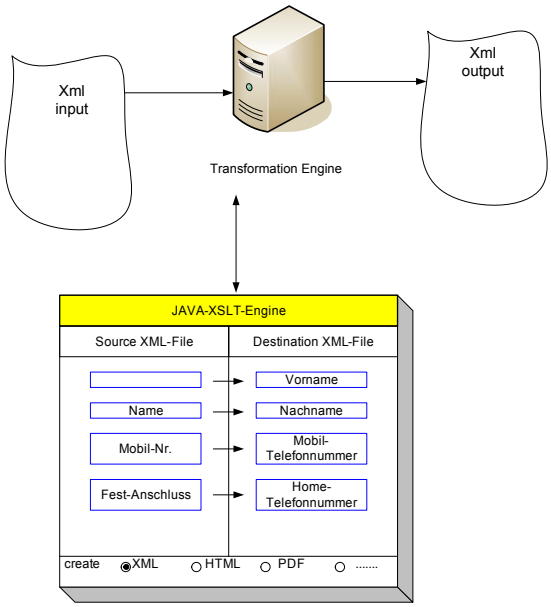


Abb. 2: Big Picture Version 1

Die Version 1 wurde den Zielen entsprechend vollkommen implementiert. Es wurde eine GUI (siehe Abb. 2: Big Picture Version 1) erzeugt, die die Möglichkeit bietet, einzelne Elemente umzubenennen. Die Ablaufstruktur sieht vor, dass ein XML-Reguest-File eingelesen (durch den *TransformationManager* mit Hilfe des *ReadHelper*) und deren Eigenschaftbeschreibungen (Tags) auf der Zuordnungsmaske auf der linken Seite (Source XML-File) angezeigt werden. Wenn die Zuordnung geändert werden muss, wird dies auf der rechten Seite (Destination XML-File) getan. Wenn die Zuordnung stimmt wird sie nicht verändert. Nach erfolgter Zuordnung (Transformation) der einzelnen Eigenschaftsbeschreibungen (Tags), muss der Button „Create“ gedrückt werden. Im Hintergrund werden die Daten aus der Tabelle (Destination XML-File) übernommen und in den *TransformationManager* gepackt. Dieser erzeugt mit Hilfe des *WriteHelper* ein neues XML-Dokument, mit den neuen Eigenschaftsbeschreibungen und den dazugehörigen Werten.

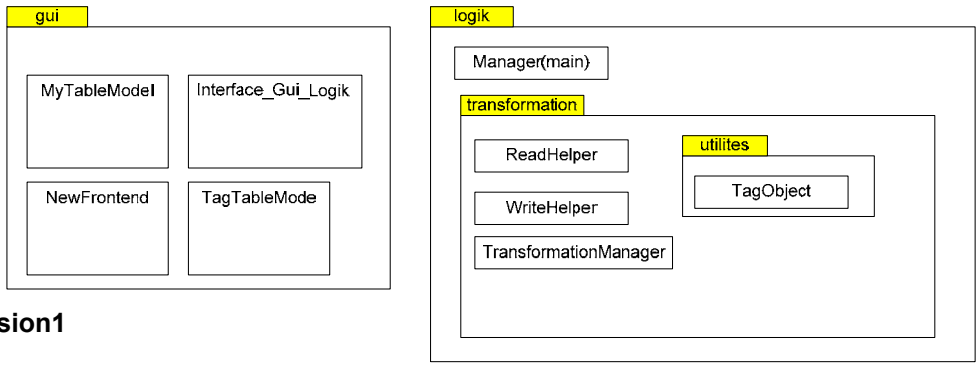


Abb. 3: Package Version1

Eigenschaften der einzelnen Objekt-Klassen: Das *TagObject* ist ein Objekt das die Eigenschaftenbeschreibung (Tag) und den dazugehörigen Wert beinhaltet. Der *ReadHelper*, hilft den *TransformationManager* die XML-Datei einzulesen und zu parsen. Dieser erzeugt ein DOMObjekt, das an den TransformationManager übergeben wird. Der *WriteHelper* ist synchron zum *ReadHelper*, wobei er keine Parserfunktion hat, sondern die Daten aus dem DOMObjekt in eine neu erzeugte XML-Datei schreibt. Der Mittelpunkt der Applikation ist der *TransformationManager*, der die Daten in Form eines DOMObjekts vom *ReadHelper* übernimmt, sie an die GUI übergibt und die geänderten Eigenschaftenbeschreibungen (Tags) im DOMObjekt transformiert. Anschließend werden sie mit Hilfe des *WriteHelper* in eine neu erzeugte XML-Datei geschrieben.

Version 2

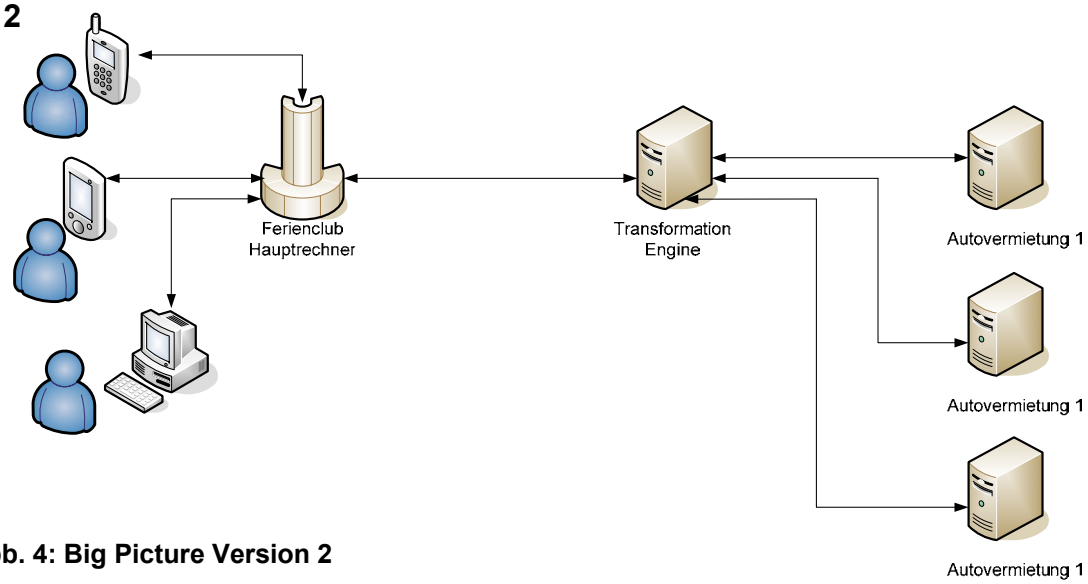


Abb. 4: Big Picture Version 2

Die Version 2 wurde den neuen Zielen entsprechend vollkommen implementiert. Die Ablaufstruktur sieht vor, dass ein XML-Reguest-File (z.B. eine Anfrage für die Mietung eines Autos bei einer Autovermietung 1) im Ferienclub eingelesen (durch den *HolidayClub* mit Hilfe des *ReadHelper*) wird. Das hier entstandene DOMObjekt mit der Datenstruktur vom Ferienclub, wird nicht direkt an die Zieladresse (hier den Autovermieter 1) geschickt, sondern alle Anfragen (DOMObjekte) werden an die TransformationEngine übergeben.

Die „Transformation Engine“ wurde als ein Web Service implementiert und bietet seine Transformationsdienste den Ferienclub und entsprechend den Zieladressen (Autovermietern) an. Die Transformation der Datenstruktur vom Ferienclub geschieht über den *TransformationManager*, der sich das Request-DOMObjekt ansieht und vom *Chooser* die dazugehörige Zuordnungstabelle holt, um die Transformation in die Datenstruktur der Zieladresse auszuführen. Nach erfolgter Transformation und den neu erzeugtem DOMObjekt, wird der Web Service der Zieladresse (hier Autovermieter 1) aufgerufen und das neue DOMObjekt übergeben.

In der Theorie würde die Zieladresse (Autovermieter 1) das ihm angepassten DOMObjekt annehmen und in seiner Datenbank nach einen dazu passenden Fahrzeug suchen. In der Implementierung ist keine Datenbank enthalten und das Problem der Suche wurde durch ein einfaches Random-Verfahren gelöst, dass ein Response-XML-Dokument 1 oder 2 per Zufall einließt (durch den *CarLender 1* mit Hilfe des *ReadHelper*). Zur Qualitätssicherung werden die Request-DOMObjekte in ein Request-XML-Dokument geschrieben (durch den *CarLender 1* mit Hilfe des *WriteHelper*).

Die Antwort geht synchron zur Anfrage über die „Transformation Engine“, in der sie transformiert wird auf die Datenstruktur des Ferienclubs, an den Ferienclub zurück.

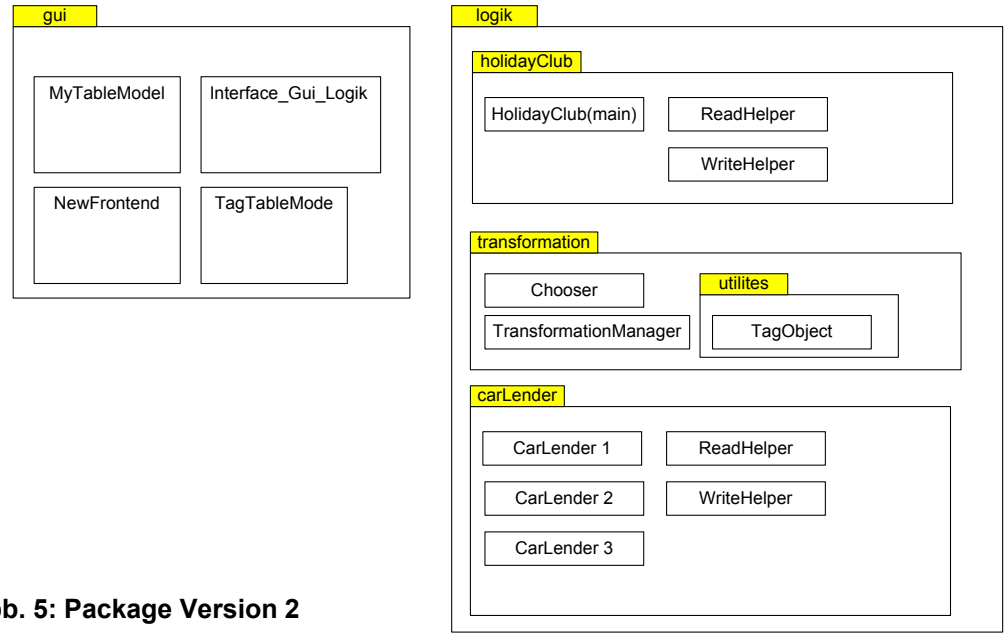


Abb. 5: Package Version 2

Eigenschaften der einzelnen Objekt-Klassen: Das *HolidayClub* Objekt hat die Aufgabe der Koordination innerhalb des Ferienclubs. Es ließt die Anfragen mit Hilfe des *ReadHelper* aus einem Request-XML-File ein und ruft den Web Service Transformation Engine auf. Nach erhalt der Antwort schreibt das *HolidayClub* Objekt mit Hilfe des *WriteHelper* das Response-DOMObjekt in ein Response-XML-File.

Das *TransformationManager* Objekt wird vom Ferienclub oder vom Autovermieter aufgerufen um eine Transformation des DOMObjekt zu gewährleisten. Für die Transformation benötigt das *TransformationManger* Objekt Information zur Zuordnung der einzelnen Eigenschaftenbeschreibungen (Tags) der beiden Partner. Diese werden vom *Chooser* Objekt zur Verfügung gestellt.

Das *CarLender* Objekt hat die Aufgabe der Koordination innerhalb des Autovermieters. Es schreibt das Request-DOMObjekt mit Hilfe des *WriteHelper* in ein Request-XML-File für eine Qualitätssicherung und speichert dieses. Nach einem Random-Verfahren, dass ein Response-XML-File auswählt. Wird dieses durch das *CarLender* Objekt mit Hilfe des *ReadHelper* eingelesen und in ein Response-DOMObjekt umgewandelt. Dieses Objekt wird an die Anfrage von der Transformation Engine zurück gegeben, die es transformiert und an den Ferienclub weiter gibt.

Die GUI wird aufgrund des *Chooser* nicht mehr verwendet, kann aber im Einzelfall wieder mit in das Projekt genommen werden.

Die Abb. 6: Overview Version 2 zeigt die Aufteilung der einzelnen Komponenten auf die Web Container. Der „Tomcat Webserver 1“ symbolisiert die SOAP-Architektur und den Process Manager 1 der den Ferienclub repräsentiert. Um unabhängig von der SOAP-Gruppe programmieren und testen zu können wurde einfach der Ferienclub anhand des Webserver 2 mit *HolidayClub* simuliert. Die Verbindung zur Transformation Engine besteht nur über einen Web Service Aufruf in Form von *getCars(aCar DomObject)*.

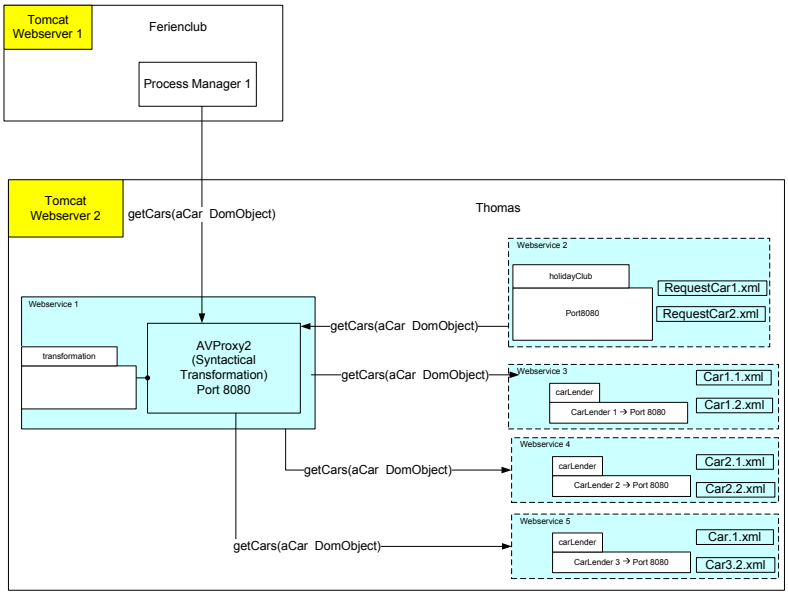


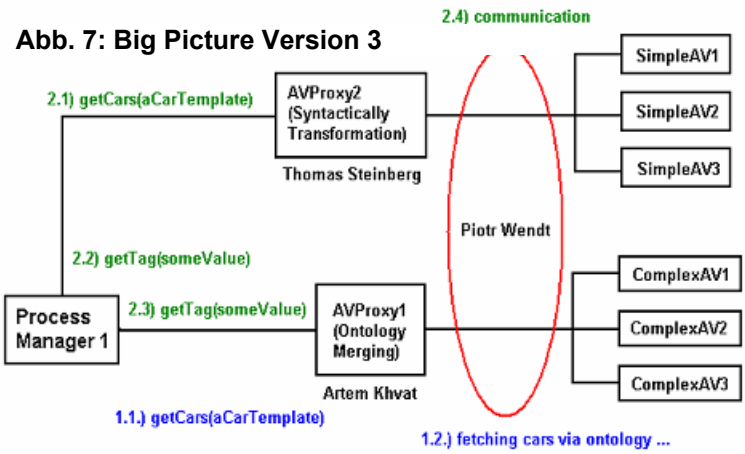
Abb. 6: Overview Version 2

Version 3

Die Version 3 wurde den Zielen entsprechend nicht implementiert, denn es stellte sich heraus dass es keinen großen Sinn hat nur eine syntaktische Information an die semantische Engine von Artem Khvat zu übergeben. Den ohne jede semantische Information könne die semantische Engine auch nur auf Übereinstimmungen der Eigenschaftsbeschreibungen (Tags) suchen und die Zuordnung machen. Dieses Verfahren ist kein anderes, was schon gemacht wird durch den *Chooser* oder durch die Zuordnungstabelle in der GUI. Somit gab es keinen wirklichen Nutzen um dieses Future zu implementieren. Des Weiteren war die Zeit auch nicht mehr ausreichend um die Version zu implementieren.

Natürlich wurden vorab Überlegungen angestellt, die folgende Ergebnisse ergaben. In der Version 3 sollte die Zusammenführung des *Process Managers 1* von der SOAP-Gruppe(SvenStegelmeier),der syntaktischen Transformation Engine(Thomas Steinberg) und der semantischen Engine(Artem Khvat) über Web Services erfolgen. Hinzukommen sollte eine Suchfunktion einzelner Web Services (Piotr Wendt).

Abb. 7: Big Picture Version 3



Die Idee war drei syntaktische (einfache) und drei semantische (komplexe)Dummy-Autovermieter als Web Service zu erzeugen. Um die Automatische Transformation zu erfüllen, sollte ein Web Service Aufruf von der syntaktischen zur semantischen Engine erfolgen 2.2) getTag(someValue).Der die Semantik in die Abfrage mit einbringen sollte. Da aber keine Semantik in der syntaktischen Transformation Engine enthalten ist, können auch keine semantischen Informationen an die semantische Engine übergeben werden. Somit besteht keine Möglichkeit einer automatischen Transformation mit Hilfe der semantischen Engine. Denn die Zuordnung der Eigenschaftsbeschreibungen (Tags) würde ähnlich wie in der syntaktischen Transformation Engine nur auf syntaktischer Ebene erfolgen.

Ausblick

In diesem Kapitel soll kurz erklärt werden, wie es zu dem Themenbereichwechsel gekommen ist und wo die neuen Erkenntnisse weiter verwendet werden. Natürlich kann nicht das gesamte Masterthema hier erläutert werden (hierzu Verweis auf das Master Thesis Outline von Thomas Steinberg), aber es wird auf die Komponenten eingegangen, die von den Vorkenntnissen des Projekts profitieren werden.

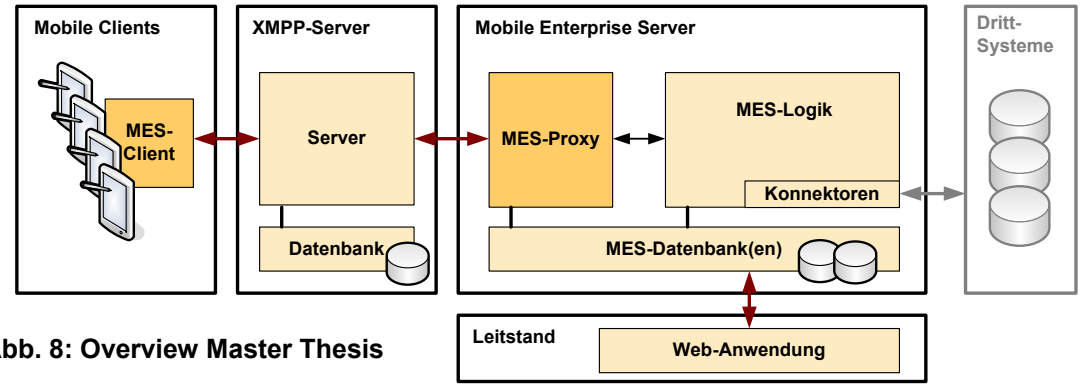


Abb. 8: Overview Master Thesis

Die Master Thesis soll eine Kommunikationsplattform werden die Aufträge an Mobile Clients verschickt, entsprechend ihrer Position. Hierzu ist eine (Mobile-) Client Architektur erforderlich, die XML-Streams empfangen kann und die Information daraus parst. Und Dritt-Systeme die als Web Service implementiert sind. Gemeint sind hier so genannte Ticket Tools für die Aufträge und Location Base Services die die Position der Mobilen Clients bestimmen können.

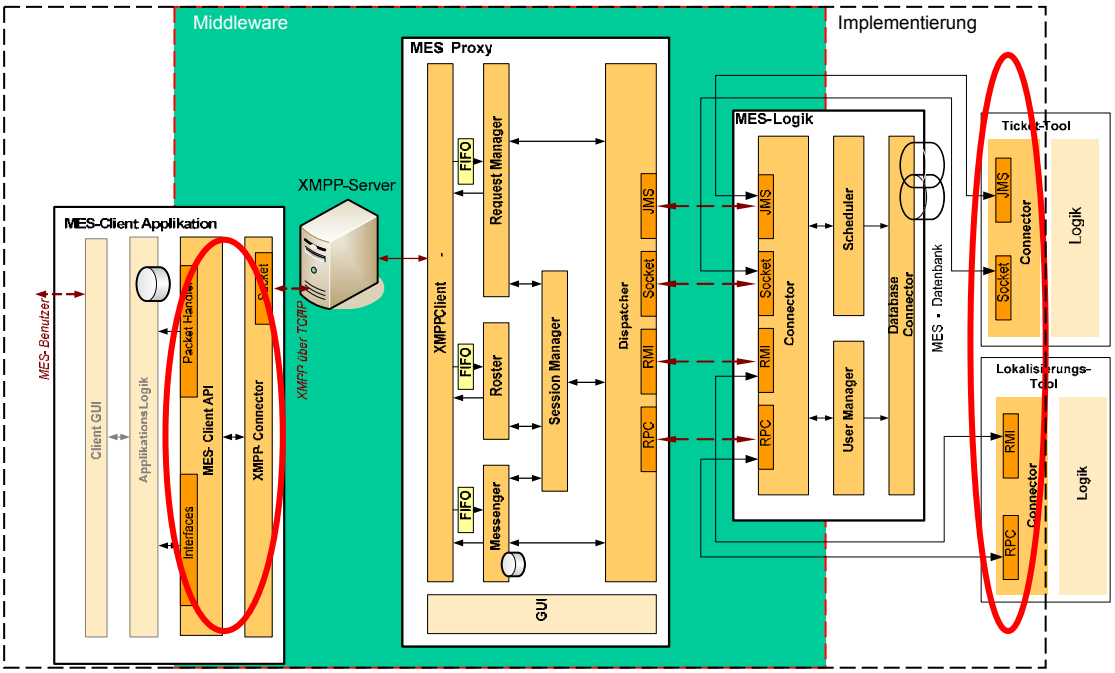


Abb. 9: Big Picture Master Thesis

Hier vielleicht noch mal detaillierter, wo auf der (Mobilen-)Clients Seite die Erkenntnisse der **Transformation** erforderlich sein werden.

Für den Aufbau der TCP/IP Verbindung und den Aufbau für den XML-Stream ist der ***XMPP***[xxii] ***Connector*** zuständig. Je nach Entwicklungsplattform kann der XMPP-Connector mit einer fertigen Bibliothek (Open-Source oder kommerziell) abgebildet werden. Um eine XMPPVerbindung aufzubauen, öffnet der Connector einen TCP/IP

Socket und initiiert einen XMLStream mit dem XMPP-Server. In dem XML-Stream werden XML-Stanzas asynchron versendet und empfangen. Der XMPP-Connector stellt für den Versand von IQ-, Messageund Presence- Stanzas Schnittstellen bereit, die von der MES-Client API genutzt werden.

Ein ***Packet Handler*****,** der zu MES[xxiii]-Client API gehört, empfängt innerhalb der MES-Client API ankommende XML-Stanzas vom XMPP-Connector und wandelt sie mit Hilfe eines XML-Parsers in Pakete um. Anschließend informiert der Packet Handler die Applikations-Logik über die Ankunft eines neuen Paketes (per Event oder Listener). Dabei werden nur die „rohen“ Pakete an die Logik übergeben. Was mit dem Paket passiert, bzw. welche Reaktion auf ein bestimmtes Paket erfolgt, muss in der Applikations-Logik festgelegt werden. Somit bleibt die MES-Client API wieder verwendbar, während die Applikations-Logik ohnehin neu auf ein kundenindividuelles Projekt angepasst werden muss.

Das Problem bei dieser Lösung wird sein, dass einmal der XMPP-Connector die XML-Daten „grob“ parsen wird und ein weiteres mal werden die XML-Daten durch den Packet Handler „fein“ geparst. Somit waren Kenntnisse erforderlich um dieses Problem zu lösen.

Die Kenntnisse von **Web Services** allgemein, werden bei den Drittsystemen gebraucht, die noch nicht weiter identifiziert wurden und während der Master Thesis genauer definiert werden.

Fazit

Was ich persönlich im Projekt gelernt habe ist hauptsächlich, dass man sein Thema während eines Projektes nicht ändern sollte, den das Wissen, dass man sich angeeignet hat(hier z.B. die Visualisierung)ist für das Projekt verloren und man verliert viel Zeit sich in ein neues Thema einzuarbeiten. Arbeitet man nicht in einer funktionierenden Gruppe ist die Arbeit schwerer und kostet mehr Zeit. Man braucht unbedingt einen Projektmanager der den Überblick über die einzelnen Themen hat und das Teilprojekt lenkt und leitet.

Allgemein zum Projekt ist zu sagen, dass mit der Fülle an Werkzeugen die XML unterstützen, lässt sich erkennen, dass XML sich wirklich schon etabliert hat. Mit Hilfe der API *JAXP 1.2* [12.] lassen sich umfangreiche Untersuchungen, Transformationen und Änderungen an XML-Dokumenten leicht bewerkstelligen. Die Anwendung von Web Services ist sehr vielfältig und fasst überall einsetzbar, was sie auch so mächtig machen. Der größte Vorteil ist aber immer noch die übergreifenden Funktionen über die Grenzen der Systeme oder der Programmiersprachen hinaus. AXIS [14.] mit seinen umfangreichen Hilfswerkzeugen ist wirklich ein mächtiges und gut funktioniertes Hilfsmittel, bei erzeugen und verwalten von Web Services.
Ein großer Vorteil ist, dass alle vorgestellten Werkzeuge auch kostenlos erhältlich sind und keinerlei Lizenzbedingungen unterliegen, was sehr förderlich für das Projekt war. Das Thema Transformation und Web Services sind nach wie vor sehr aktuell. Anhand des Ausblicks sieht man, dass diese Themen mich auch während der Master Thesis weiter beschäftigen werden.

Quellen

[1.] Java und XML Verarbeitung, Auswertung und Transformation von XML-Dokumenten und deren Verwendung über Java RRZN Universität Hannover
[2.] XSLT : [XML-Dokumente transformieren] / Doug Tidwell.
ISBN: 3-89721-292-7
[3.] XSLT - Anwendung und Referenz : XML-Transformationen, XPath, Einsatz mit Java, JSP und ASP / Steven Holzner. Übers. von Reader Translations
ISBN: 3-8272-6260-7
[4.] Web Services: Die Standards von T.Hauser & U.M.Löwer Galileo Computing
ISBN : 3-89842-393-X
[5.] Service-orientierte Architekturen mit Web Services Wolfgang Dostal
ISBN : 3-8274-1457-1
[6.] Building Web services with Java : XML, SOAP, WSDL, and UDDI / Steve Graham
ISBN: 0-672-32641-8
[7.] Verteilte Systeme und Anwendungen : Architekturkonzepte, Standards und Middleware-Technologien / Ulrike Hammerschall
ISBN3-8273-7096-5
[8.] Understanding SOA with Web services / Eric Newcomer; Greg Lomow
ISBN: 0-321-18086-0
[9.] http://java.sun.com/webservices/containers/tomcat_for_JWSDP_1_5.html
[10.] http://java.sun.com/webservices/downloads/webservicespack.html
[11.] http://www.apache.org/dist/xml/xerces-j/
[12.] http://java.sun.com/xml/downloads/javaxmlpack.html
[13.] http://www.eclipse.org/downloads/
[14.] http://ws.apache.org/axis/
[15.] http://www.oio.de/axis-soap.htm

Glossar

[i] XSL	:	Extensible Stylesheet Language
[ii] PDA	:	Personal Digital Assistant
[iii] Parser	:	Analysieren von XML Dokumenten
[iv] DTD	:	Document Type Definition
[v] SAX	:	Simple API for XML
[vi] DOM	:	Document Object Model
vii Xerces 2	:	DOM-Parser
viii XSLT	:	Extensible Stylesheet Language Transformation
ix Stylesheet	:	Formatvorlage
x Xalan	:	Apache Group XSLT Prozessor
xi XPath	:	XML Path Language
xii FO	:	Formating Objects
xiii UDDI	:	Universal Description, Discovery and Integration
xiv WSDL	:	Web Services Description Language
xv XML-RPC	:	XML-Remote Procedure Call
xvi AXIS	:	Apache eXtensible Interaction System
xvii Transformation Engine	:	Transformation Maschine (syntaktisch)
xviii BI-Team	:	Business Intelligence Gruppe
xix Semantische Engine	:	Transformation/Merge Maschine (semantisch)
xx SOAP-Gruppe	:	Simple Object Access Protocol Gruppe
xxi SOAP-Architektur	:	Simple Object Access Protocol
xxii XMPP	:	Extensible Messaging and Presence Protocol
xxiii MES	:	Mobile Enterprise Server (Master Thesis)