Konzeption und Implementierung von Web Services zur Integration des Webshops der LUBW in andere Fachinformationssysteme des Umweltinformationssystems Baden-Württemberg

DIPLOMARBEIT

für die Prüfung zum

Diplom-Ingenieur (BA)

Studiengang Informationstechnik

an der Berufsakademie Karlsruhe

von

Marco Schulze

September 2006

Bearbeitungszeitraum	06.06.2006 bis 04.09.2006
Kurs	TIT2003
Ausbildungsfirma	Landesanstalt für Umwelt, Messungen und Naturschutz Baden-Württemberg (LUBW) Karlsruhe
Gutachter der Ausbildungsfirma	Dipl. Inform. Renate Ebel
Gutachter der Studienakademie	Dr. Clemens Düpmeier

Versicherung

„Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig und nur unter Nutzung der angegebenen Literatur und Hilfsmittel angefertigt habe. Wörtlich übernommene Sätze und Satzteile sind als Zitate belegt, andere Anlehnungen hinsichtlich Aussage und Umfang unter den Quellenangaben kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen und ist nicht veröffentlicht."

Kurzfassung

Konzeption und Implementierung von Web Services
zur Integration des Webshops der LUBW
in andere Fachinformationssysteme
des Umweltinformationssystems Baden-Württemberg

Im Umweltinformationssystem Baden-Württemberg gibt es eine große Zahl von Fachdokumenten, die in unterschiedlichen Systemen verwaltet werden. Ein Vorschlag für eine künftige Integration aller Fachdokumentbestände liegt in einem Grobkonzept vor. In dieser Arbeit wird nun am Beispiel eines Fachsystems, dem Webshop der LUBW, das Grobkonzept verfeinert und umgesetzt, sowie ein Web Service implementiert und getestet. Es werden am Anfang die beteiligten Systeme betrachtet und erklärt. Weiterhin wird das Feinkonzept für das vorgegebene Beispiel unter der Berücksichtigung mehrerer Lösungsstrategien erstellt. Die beste Strategie wird implementiert und durch einen Testclient getestet. Der Test zeigt, dass die Schnittstellen des implementierten Web Services funktionieren und die geforderten Daten in der richtigen Form liefern. Damit wird nachgewiesen, dass die Anbindung des LUBW Webshops in eine übergreifende Integrationskomponente möglich ist. Damit haben UIS-Entwickler künftig mehr Möglichkeiten, um Fachdokumente des Shops in ihrer Fachsystemumgebung nutzerfreundlich zu integrieren.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Abkürzungsverzeichnis

RMI	Remote Procedure Call
FDB	Fachdokumentenbrowser
WSWS	WebShop Web Service
CMS	Content Management System
LUBW	Landesanstalt für Umwelt, Messungen und Naturschutz Baden-Württemberg
PortalU	Umweltportal Deutschland
IDE	Abkürzung IDE, von engl. integrated development environment, auch integrated design environment
OOP	objektorientierte Programmierung

IV. Tabellenverzeichnis

1. Einleitung

Die Landesanstalt für Umwelt, Messungen und Naturschutz (LUBW) ist seit 10 Jahren im Internet präsent (früher als Landesanstalt für Umweltschutz – LfU). Innerhalb dieser 10 Jahre hat sich ein riesiger Bestand von ca. 10 000 Fachdokumenten wie Umweltberichte, Handlungsanweisungen für die Umweltverwaltung bis hin zu offiziellen Publikationen angesammelt. Diese Situation existiert auch bei anderen Fachsystemen im gesamten Umweltinformationssystem Baden-Württemberg (UIS).

Bisher wurden die Fachdokumente durch die verschiedenen Fachsysteme mit unterschiedlichen Zugängen und Suchfunktionen verwaltet. Dadurch kam es oft zu Überschneidungen in den Inhalten und den Funktionen. Vor allem bei der Ablage von Fachdokumenten kam es immer wieder zu mehrfachen Speicherungen, was zu Problemen und Missverständnissen in der Aktualisierung und dem Nachweis der neuesten Dokumentenversion führte. Aufgrund der teilweisen Vernetzung der einzelnen Systeme und der Kommunikation zwischen ihnen haben sich die Überschneidungen verringert.

Der nächste Schritt ist eine übergreifende Software-Komponente welche in der Lage ist, einen einheitlichen Zugang zu den Fachdokumenten aller Fachsysteme zu schaffen. Durch diese Komponente können die Dokumentenbestände zwischen den Fachsystemen koordiniert und überwacht werden. Um dies zu erreichen, muss die Kommunikation über standardisierte Schnittstellen erfolgen. Der Fachdokumentenbrowser, den das Forschungszentrum Karlsruhe (FZK) im Auftrag der LUBW entwickelt, soll diese übergreifende Komponente darstellen, welche die Fachdokumentenbestände koordiniert und überwacht. Über den Fachdokumentenbrowser soll u. a. auch auf die Publikationen des LUBW Shops zugegriffen werden können.

In den nachfolgenden Kapiteln wird hierzu ein Feinkonzept für die Integration der Publikationen des LUBW-Shops aus dem Content Management System (CMS) der LUBW mit dem Fachdokumentenbrowser (FDB) erstellt. Dabei werden verschiedene Lösungsstrategien aufgezeigt und die Vor- und Nachteile diskutiert. Am Ende wird die beste Lösung in das Feinkonzept aufgenommen. Weiterhin werden die Schnittstellen für den FDB und die Implementierung dieser erläutert.

Schließlich werden verschiedene Testmöglichkeiten für die implementierten Schnittstellen aufgezeigt.

2. Aufgabenstellung und Rahmenbedingungen

In diesem Kapitel werden Aufgabenstellung und Vorgaben erläutert. Anschließend werden wichtige Begriffe erklärt. In den Rahmenbedingungen werden schließlich die beteiligten Systeme und Schnittstellen vorgestellt, soweit sie für die vorliegende Arbeit relevant sind.

2.1. Aufgabenstellung

Es soll ein Feinkonzept für die Integration der angebotenen Dokumente des Webshops der LUBW mit dem Fachdokumentenbrowser (FDB), den das FZK entwickelt, erstellt werden. Für diese Aufgabe ist es in einem ersten Schritt nötig, die Benutzungsschnittstellen klar zu definieren und zu beschreiben. Weiterhin werden Lösungsstrategien zur Implementierung der Integrationsschnittstelle diskutiert. Die beste Strategie soll dabei aufgezeigt und anschließend implementiert und getestet werden.

2.2. Vorgaben

Jedes Fachsystem muss für den FDB zwei Web Service Schnittstellen bereitstellen. Diese Schnittstellen müssen programmiersprachenunabhängig angesprochen werden können und über das Internet erreichbar sein. Die Schnittstellen sollen den Fachdokumentenbrowser in die Lage versetzen, alle Id's der Fachdokumente des jeweiligen Fachsystems zu ermitteln, sowie die Metadaten eines bestimmten Fachdokumentes in die FDB eigene Metadatenbank speichern zu können.
Weiter Anforderungen vom FDB sind zum Zeitpunkt dieser Diplomarbeit nicht gefordert. Die Schnittstellen sollen sich auf die genannten Funktionen beschränken.
Von Seiten der LUBW sind die Web Service Schnittstellen in Java zu implementieren. Der Grund liegt in dem eingesetzten CMS WebGenesis[1] und dessen Programmierumgebung. Das CMS ist in Java geschrieben und bietet nur für Java Clients eine optimale Möglichkeit, die WebGenesis internen Schnittstellen zu nutzen. Weiterhin wird WebGenesis in der LUBW auf einer Linux Server Architektur eingesetzt, daher wird Java als plattformunabhängige Sprache favorisiert.
Der Test der implementierten Lösung soll so gestaltet werden, dass verifiziert wird, dass der bereitgestellte Web Service unabhängig von einer bestimmten Programmiersprache (insbesondere verschieden von Java) angesprochen und benutzt werden kann. Zum Test der Funktionalität des Web Services sollen die angeforderten Daten aus dem Fachsystem prototypisch verarbeitet werden. Diese Verarbeitung konzentriert sich auf die Ausgabe der Id's bzw. der Metadaten, so dass hierbei die zwei wesentlichen Nutzungsinterfaces getestet werden.

[1] http://www.webgenesis.de

2.3. Grundlagen und Begriffserklärung

2.3.1. Fachdokument

Jedes Fachsystem des UIS besitzt Dokumente, welche für Fachleute bzw. der Allgemeinheit zur Verfügung gestellt werden.
Ein Fachdokument wird im Kontext dieser Arbeit wie folgt definiert:

- liegt in geschlossener Form als Datei vor, z.B. als PDF, DOC, XLS, HTML, Archiv-Dateien (Zip)
- wenn im engeren Sinne verstanden, ist der Bericht in einem Textformat gespeichert oder kann auch im weiterem Sinne als Grafik-, Audio-, oder Videodatei vorliegen
- enthält Texte, Sachdaten, Grafiken; Audio- und Videodateien können auch Bestandteil eines Fachdokumentes sein
- enthält Fachinformationen
- hat längerfristigen Charakter (Weitergabe in Dateiform ist möglich und beabsichtigt) und kann über Metadaten beschrieben werden

2.3.2. Web Service

Die Idee hinter Web Services ist nicht neu: Übertragungsstandards sollen dafür sorgen, dass Daten und Funktionalität zwischen Kommunikationspartnern ausgetauscht werden können.
Wikipedia definiert einen Web Service[2] als eine Software-Anwendung (besser wäre hier Softwarekomponente), die mit einem URI[3] eindeutig identifizierbar ist und deren Schnittstellen als XML-Artefakte definiert, beschrieben und gefunden werden können.
Eines der wichtigsten Einsatzgebiete von Web Services ist die Integration von Geschäftsprozessen. Hier bietet die Technologie die Möglichkeit, verteilte Anwendungen über standardisierte Schnittstellen zu verknüpfen und Daten im XML-Format auszutauschen.
Das Internet wird bevorzugt als Kommunikationsmedium für Web Service Applikationen genutzt. Das hat zum einen den Vorteil, dass vorhandene Infrastrukturen mit bekannten und bewährten Protokollen genutzt werden können und zum anderen das Internet von nahezu überall zugänglich ist.
Feste Voraussetzungen für einen Web Service sind, dass sie von den verschiedensten Geräten, wie PCs, Handys oder PDAs, genutzt werden können. Sie müssen unabhängig vom Betriebssystem und der Programmiersprache angesprochen werden können. Leider impliziert die unabhängige Nutzung eines Web Services von der Programmiersprache nicht die optimale Nutzung des Web Services, siehe WebGenesis. Hier kann zwar die Web Service Schnittstelle von verschiedenen Programmiersprachen angesprochen, aber nicht optimal genutzt werden.

[2] Nachgeschlagen in Wikipedia unter http://de.wikipedia.org/wiki/Web_Service am 01.08.06
[3] Uniform Resource Identifier

Web Services sind nicht eine Technologie, die einheitlich in einem Dokument geregelt ist. Vielmehr basieren Web Services auf verschiedenen Standards und Spezifikationen, die im Zusammenspiel diese Technologie bilden. Drei XML-Technologien bilden die Grundlage von Web Services: SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language) und UDDI (Universal Description, Discovery and Integration).
Die Web Service-Architektur basiert auf der Interaktion zwischen drei Rollen: Serviceanbieter, Servicekonsument und Servicebroker.

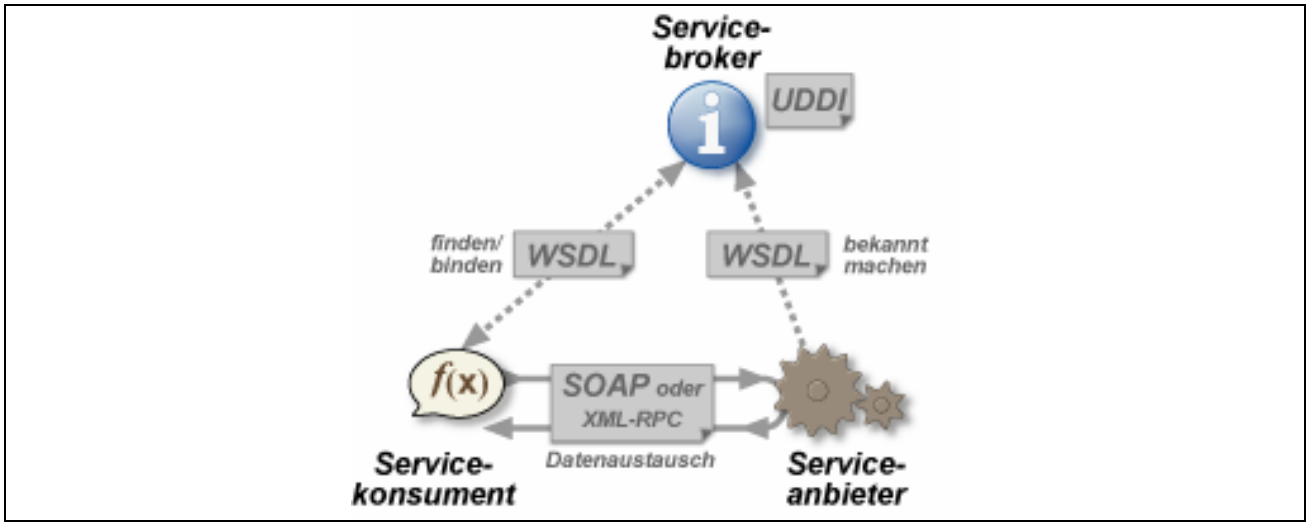

Abbildung 1: Web Service Roles

Serviceanbieter - implementiert einen Service und stellt diesen Service im Netzwerk zur Verfügung. Dieser Service ist im Netzwerk adressierbar und besitzt eine klar definierte Schnittstelle.
Servicekonsument - sind alle möglichen Kunden des Services. Falls die Schnittstelle und die Adresse des Services nicht bekannt sind, muss der Kunde zunächst durch eine entsprechende Anfrage beim Servicebroker die notwendigen Informationen in Erfahrung bringen. Danach kann er den Service aufrufen, in dem er einen Request in einem geeigneten Format an den Service schickt und den erwünschten Service als Antwort darauf erhält.
Servicebroker - ein zentrales, logisches Verzeichnis für Service-Registrierung und -Discovery. Der Servicebroker ist ein optionales Element und kommt nur dann als Vermittler ins Spiel, wenn sich Anbieter und Konsument nicht direkt kennen.

Als Standard für die Beschreibung eines Web Services hat sich die Web Service Description Language (WSDL) durchgesetzt.
Die WSDL ist eine formale (maschinengestützt lesbare) Beschreibung der Aufrufschnittstelle eines Web Services, d.h. der Servicekonsument kann die WSDL Beschreibung eines Services abrufen und kennt mit ihrer Hilfe die Methoden des Web Services. In der Beschreibungsdatei wird der ei-

gentliche SOAP-Service referenziert. Die WSDL-Datei erleichtert somit die Identifikation der Servicedienste und ermöglicht eine automatische Integration dieser.

Eine WSDL-Datei besteht wie ein XML-Dokument aus einer Vielzahl an Elementen. Die Grundelemente sind:

- *definitions* ist das Wurzelelement. Hier werden der Name des Services, sowie dessen Namespace und verwendete Standards definiert.
- *types* enthält Datentyp-Definitionen.
- *message* enthält die Informationen über die Daten in einer Nachricht des Web Services. Bei mehr als einer enthaltenden Nachricht, z.B. Anfrage und Antwort, gibt es mehrere message-Elemente.
- *portType* enthält Angaben über die Methoden des Web Services.
- *binding* gibt an, mit welchem Protokoll und in welchem Format die im portType definierten Methoden abgearbeitet werden.
- *service* definiert die Endpunkte eines Dienstes.

Universal Description, Discovery and Integration of Web Services (UDDI) ist der Standard für Web Service-Verzeichnisse im Internet. In diesem Verzeichnisdienst stehen Anbieter mit ihren Daten und angebotenen Diensten.
Es gibt drei verschiedene Arten von Informationen in UDDI. Die so genannten "White Pages", einer Art Telefonbuch, die "Yellow Pages", also die elektronische Entsprechung der gelben Seiten, und die "Green Pages".

- **White Pages**
  - Namensregister, sortiert nach Namen
  - Auflistung der Anbieter mit allen Detailangaben
  - Kontaktinformationen (Telefon, Telefax,...)
- **Yellow Pages**
  - Branchenverzeichnis
  - Spezifische Suche gemäß verschiedener Attribute (Ort, Dienstart,...)
  - Verweist auf White Pages
- **Green Pages**
  - Informationen über Geschäftsmodell des Anbieters
  - Technische Details zu den angebotenen Web Services
  - Auskunft über Geschäftsprozesse

UDDI hat sich im Internet nicht richtig durchgesetzt. Ein Grund dafür könnte das schlechte Vertrauensverhältnis zu den Konkurrenten sein. Allerdings ist UDDI im Intranet sehr interessant. Insbesondere das automatische Finden von angebotenen Web Services. Ein anderer Grund ist, dass

mit Hilfe von UDDI-Verzeichnissen nicht nur Web Services beschrieben werden können, sondern auch alle anderen Webdienste, wie z.B. Dienste, die über parametrisierte URL's aufgerufen werden können.

Für die Kommunikation zwischen den Applikationen und dem Web Service wird SOAP verwendet.
SOAP war ursprünglich eine Abkürzung für Simple Object Access Protocol. Da diese Bezeichnung zu Missverständnissen führte, ist SOAP ab Version 1.2 keine Abkürzung mehr.
SOAP ist ein Protokoll, mit dessen Hilfe Daten für entfernte Funktionsaufrufe in Form von Nachrichten über Internetprotokolle wie HTTP oder SMTP zwischen Systemen ausgetauscht werden können. Dabei sind die Nachrichten XML-konform und haben einen bestimmten Aufbau.
SOAP ist kein Transportprotokoll. Zum Transport von SOAP-Nachrichten wird ein separates Transportprotokoll benötigt. Meistens kommt dabei HTTP zum Einsatz, je nach Anwendung ist aber auch SMTP oder FTP möglich.
Der Aufbau einer SOAP Nachricht wird in der folgenden Abbildung aufgezeigt.

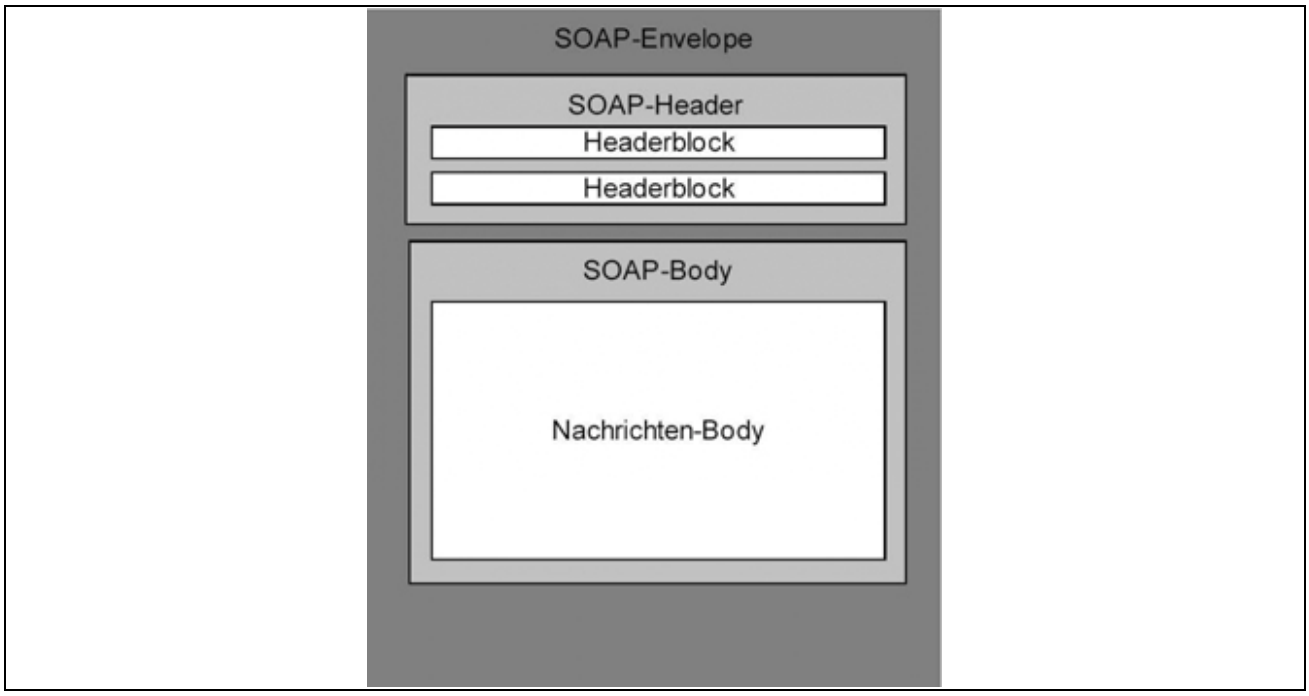

Abbildung 2: Aufbau einer SOAP Nachricht

SOAP-Envelope: Dies ist das Wurzelelement einer SOAP-Nachricht und umschließt alle anderen Elemente wie ein Umschlag.

SOAP-Header: Optional. Enthält Verwaltungsinformationen bzw. zusätzliche Daten wie zum Beispiel Daten, die für Zwischenstationen (Intermediaries) bestimmt sind.

SOAP-Body: Enthält die eigentlichen Nutzdaten. Bei einem Request sind dies die erforderlichen Parameter, und bei der Antwort die Rückgabewerte.

SOAP-Fault: Fehler. Dieses Element ist nur dann in der Nachricht im Body enthalten, wenn ein Fehler aufgetreten ist.

2.4. Rahmenbedingungen

2.4.1. Webshop der LUBW

Ein Fachsystem stellt spezielle Umweltfachinformationen bereit, welche nach bestimmten Kategorien unterteilt sind. Diese Kategorien umfassen die gesamte Umweltthematik, z.B. Klima, Luft, Boden, Wasser, Abfall, Umweltschutz und viele andere. Im Kontext dieser Diplomarbeit werden nur Fachsysteme im UIS Baden-Württemberg betrachtet, die eine Vielzahl von Fachdokumenten verwalten. Von diesen wird speziell der Webshop der LUBW genauer behandelt.
Im Webshop (siehe Abbildung 3 und 4) werden Fachdokumente der LUBW für die Öffentlichkeit bereitgestellt. Diese speziellen Fachdokumente haben den offiziellen Status einer Publikation. Die Fachdokumente bzw. Publikationen sind im Webshop in Umweltkategorien unterteilt, welche eine Untermenge der Umweltthemen vom Umweltportal Deutschland (PortalU) sind. Die Publikationen werden über einen Mitarbeiter der Bibliothek von der LUBW erfasst, beschrieben und im Shop eingestellt. Sie liegen als PDF-Dateien, Broschüren oder als Links auf HTML-Seiten vor. Die Dokumente können kostenpflichtig bzw. kostenlos erworben werden. Die Problematik ist, dass die Publikationen nur über die LUBW Internetseiten aufgerufen werden können, und es momentan keine Möglichkeit gibt, diese anderen Systemen für eine Integration in deren Umgebung zur Verfügung zu stellen.

Die Publikationen im Shop der LUBW sind in verschiedene Umweltkategorien unterteilt. Die Themen lauten wie folgt:

- Allgemeine Umweltfragen
- Abfall
- Altlasten
- Betrieblicher Umweltschutz
- Boden
- Chemikalien und Arbeitsschutz
- Elektromagnetische Felder
- Klima
- Lärm
- Luft
- Natur und Landschaft
- Radioaktivität
- Wasser
- Lokale Agenda 21

Diese Themen sind eine Untermenge der Themen im PortalU. Daher muss eine Zuordnung der LUBW Kategorien zu den des PortalU erstellt werden. Die Themenzuordnung ist nicht festgelegt und kann sich im Laufe der Zeit ändern. Die Umweltkategorien entsprechen den obersten Verzeichniseinträgen des Shops der LUBW, siehe Abbildung 3.

Abbildung 3: Webshop der LUBW – Einstiegsseite

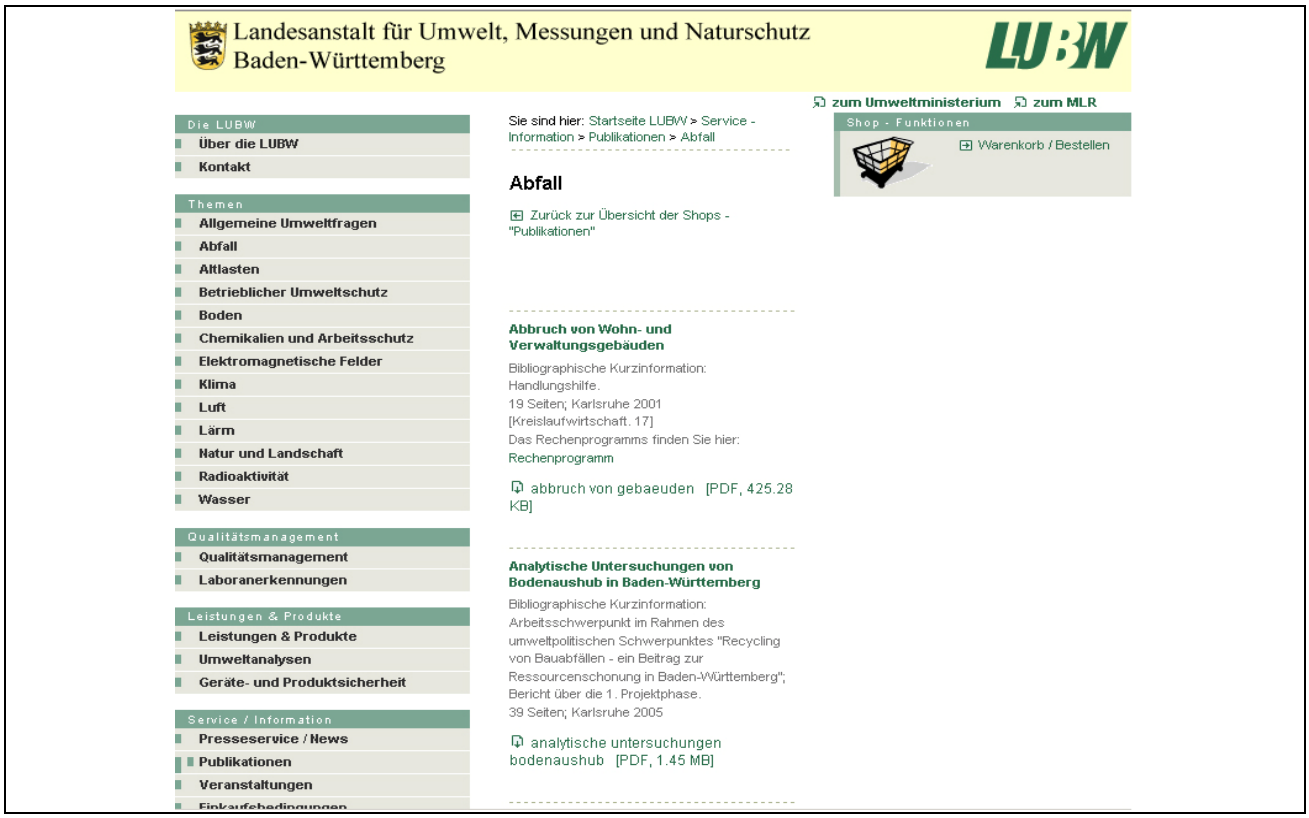

Abbildung 4: Webshop der LUBW – Auflistung der Publikationen zum Thema Abfall

Der Webshop ist mit dem CMS WebGenesis implementiert, das in der LUBW für das Internetangebot benutzt wird. Es ist deshalb sinnvoll, die Schnittstellen genauer zu betrachten, die WebGenesis bereits standardmäßig zur Verfügung stellt.

2.4.2. Web Service Schnittstellen von WebGenesis

WebGenesis besitzt ab Version 7.00 generische Web Service Schnittstellen, mit deren Hilfe man einen großen Teil der internen Funktionalität des Systems über Web Services aufrufen kann. D.h. man kann sich z.B. in das System einloggen bzw. ausloggen, es können WebGenesis-Befehle ausgeführt oder Dateien hoch und herunter geladen werden. Diese Web Service Schnittstellen bieten alle mit WebGenesis realisierten Systeme im UIS an. Insbesondere lässt sich mit dem generischen Web Service der WebGenesis Systeme auch auf die internen Publikationen im Webshop der LUBW zugreifen. Laut Entwicklerhandbuch von WebGenesis können die Web Service Schnittstellen nicht nur über Java, sondern auch von anderen Programmiersprachen wie C# oder .NET genutzt werden. In der Praxis hat sich jedoch gezeigt, dass über Nicht-Java Clients nicht alle der implementierten Funktionalitäten aufgerufen werden können.
Es wird daher empfohlen, die Clients in Java zu programmieren, damit hierdurch eine optimale Nutzung der Schnittstellen gewährleistet wird. Der in dieser Arbeit vorgestellte Web Service ist daher als Java Web Service Client von WebGenesis implementiert, soll aber eine Nutzung durch Clients in anderen Programmiersprachen ermöglichen.

WebGenesis stellt zwei Kommunikationsmöglichkeiten bereit. Das ist zum einen eine RMI[4] und zum anderen eine SOAP Schnittstelle.
Die Kommunikation über RMI ist schneller als die über SOAP. Allerdings muss für RMI ein extra Port freigeschaltet werden. Weiterhin ist RMI eine Java spezifische Implementierung eines Remote Procedure Calls (RPC) und kann damit nur von Java selbst benutzt werden.
SOAP hingegen ist ein Protokoll, welches von vielen Programmiersprachen verwendet werden kann. Da SOAP unter anderem HTTP als Transportprotokoll benutzt, kann die Kommunikation über den Port 80 laufen, d.h. es müssen keine Anpassungen an der Firewall gemacht werden.

2.4.3. Fachdokumentenbrowser

Der Fachdokumentenbrowser ist zum Zeitpunkt der Diplomarbeit noch nicht implementiert und nur als Grobkonzept verfügbar. Dieses Grobkonzept wird in der Studie „Fachdokumentenmanagement im Umweltinformationssystem Baden-Württemberg“ beschrieben. Der FDB soll eine eigene Datenbank besitzen in die er die Metadaten und die Id’s aller Fachdokumente der einzelnen Fachsysteme speichern kann. Über den FDB sollen die Fachleute bzw. der allgemeine Benutzer die Möglichkeit haben, den gesamten Bestand an Fachinformationen zu erhalten ohne explizit in den einzelnen Fachsystemen suchen zu müssen.

2.4.4. Zielarchitektur

Im Vorfeld wurden die einzelnen Systeme, wie der Fachdokumentenbrowser und die Fachsysteme, speziell der Webshop der LUBW, vorgestellt. Diese Systeme sind noch eigenständige Systeme. Die Fachsysteme stellen Fachdokumente bereit. Durch den FDB sollen diese Dokumentbestände auf der Metadatenebene vereinigt werden. Um dies zu erreichen müssen die Fachsysteme mit dem Fachdokumentenbrowser kommunizieren können. Diese Kommunikation soll über Web Service Schnittstellen erfolgen.
Der in dieser Arbeit vorgestellte Web Service soll über Web Service Schnittstelle verfügen, welche von WebGenesis Listen der Dokumente im Webshop, sowie die Metadatenbeschreibung einzelner Publikationen zur Verfügung stellen und damit auch eine Integration der Webshop Dokumente in anderen Systemen (vor allem im Fachdokumentenbrowser) möglich machen. Hierbei müssen die Metadaten von Dokumenten im Webshop (wie z.B. die Kategorisierung) geeignet auf die Metadaten, wie sie der Fachdokumentenbrowser benötigt, abgestimmt werden. Eine zentrale Aufgabe dieser Arbeit ist es daher, die existierenden Metadaten der Dokumente im Webshop auf geeignete Art und Weise auf die Metadaten abzubilden, die der FDB benötigt.

[4] Remote Method Invocation ist eine Java-eigene Remote Procedure Call (RPC) Art

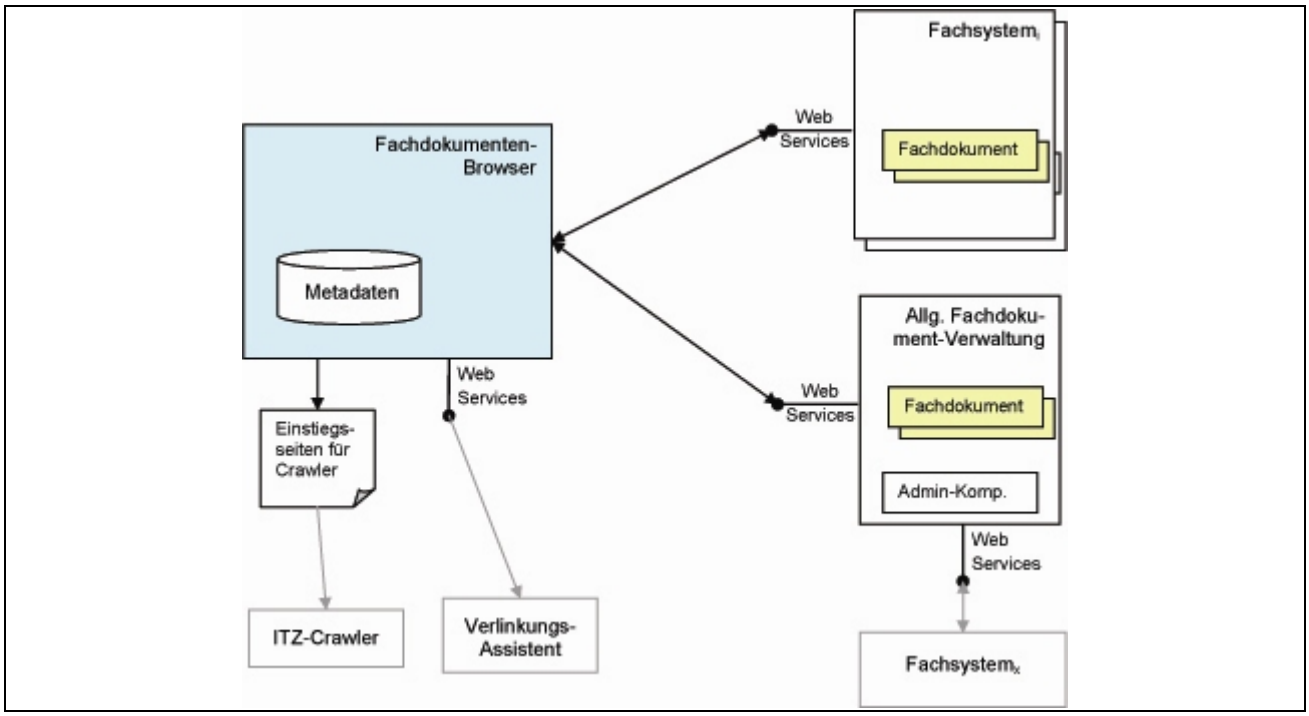

Abbildung 5: Grobkonzept des Aufbaus vom Fachdokumentenbrowser

3. Konzept

3.1. Lösungsansätze und Analyse

Diese Diplomarbeit soll zeigen, dass das Grobkonzept aus der Studie „Fachdokumentenmanagement im Umweltinformationssystem Baden-Württemberg“ machbar ist. Dazu wird ein kleines Modell bestehend aus dem WebGenesis-System, dem implementierten Web Service und der Testkomponente konzipiert, welches die Architektur aus dem Grobkonzept widerspiegelt. Dabei übernehmen WebGenesis und der Web Service die Funktion der Fachsysteme mit ihren Web Service Schnittstellen und die Testkomponente die Funktion des Fachdokumentenbrowsers. Es wird hier bewusst auf die Verwendung des Web Services durch andere Systeme verzichtet und die damit möglichen zusätzlichen Funktionsanforderungen, z.B. „gib alle Id’s von Dokumenten, welche sich im Zeitraum X geändert haben“. Der Grund ist, dass diese Diplomarbeit speziell auf die Kooperation zwischen dem FDB, mit den Anforderungen zum Zeitpunkt der Diplomarbeit und dem Web Service ausgelegt ist.

3.1.1. Lösungen für den Web Service

In diesem Abschnitt werden mehrere Lösungsansätze vorgestellt und diskutiert. Es werden die Vor- und Nachteile jeder Lösung gegenüber gestellt und bewertet.
Im ersten Lösungsansatz wird der WebGenesis eigene Web Service als direkte Schnittstelle für den FDB betrachtet. Der WebGenesis Web Service bietet einen „Low Level“ Zugriff auf die Basisfunktionalitäten eines WebGenesis-Systems an. Diese sind z.B. das Login, das Logout, das Up- und Downloaden von Dateien und das Ausführen von Befehlen der WebGenesis Kommandoprozessoren.
Die WebGensis Web-Service API stellt aber keinerlei „höherwertige“ Funktionalitäten bereit, die es dem FDB ermöglichen würde, direkt die Id’s der Dokumente des Shops bzw. die Metadaten eines bestimmten Dokumentes zu erhalten. Die Entwickler des FDB müssten hiefür Kenntnisse von der WebGenesis-Befehlssyntax und dem WebGenesis-System an sich haben, um die gewünschten Daten zu erhalten.
Der Vorteil bei diesem Ansatz liegt darin, dass die eigentliche Arbeit die FDB-Entwickler und nicht die Entwickler der Fachsysteme, welche auf WebGenesis basieren, haben. Die WebGenesis-Entwickler müssen nur die Schnittstellen, auch bei neuen Systemversionen, stabil beibehalten. Diese Lösung ist nicht akzeptabel, da der FDB nicht für jedes System eigene Spezialanfragen erstellen kann.

Ein anderer Lösungsansatz wäre, wenn die geforderten Methoden, „gib alle Id’s der Dokumente aus dem Shop“ und „gib alle Metadaten eines bestimmten Dokumentes“, direkt von WebGenesis

angeboten würden. Dazu muss aber am System direkt entwickelt werden. Für diesen Ansatz muss der Programmierer an einer Entwicklerschulung von WebGenesis teilgenommen haben, damit er das System kennt. Zusätzlich besteht die Gefahr, dass bei einem Versionswechsel die entwickelten Zusatzfunktionen nur noch teilweise bzw. gar nicht mehr funktionieren. Ein weiterer Nachteil ist das Anpassen bzw. Weiterentwickeln des Web Services. Es müssten jedes Mal mit den WebGenesis-Entwicklern Absprachen gehalten werden, damit die neuen Funktionen auch in der nächsten Version enthalten sind bzw. müssen die Zusatzfunktionen in das WebGenesis-System als Standard mit aufgenommen werden.
Hier liegt der Vorteil darin, dass die Entwicklung der geforderten Methoden nicht die FDB- bzw. die Entwickler aus dem Bereich des Fachsystems, z.B. LUBW-Mitarbeiter, selber durchführen müssten, sondern dem WebGenesis-Team überlassen könnten.
Diese Lösung wird auch nicht als ideal angesehen, da die Weiterentwicklung der Schnittstellen direkt über die WebGenesis-Entwickler den Zeitrahmen der Diplomarbeit überschreiten würde und sich zusätzliche Mehrkosten ergeben könnten.

Der letzte Lösungsansatz, der hier vorgestellt wird, beinhaltet die Entwicklung eines eigenen Web Services, welcher zwischen dem FDB und dem WebGenesis-System steht. Dieser Web Service bietet die Schnittstellen für den Fachdokumentenbrowser wie gefordert an. Zusätzlich kommuniziert er mit WebGenesis, um die vom FDB gewünschten Daten zu erhalten und gibt sie dem Fachdokumentenbrowser weiter.
Der Vorteil ist hierbei, dass weder die FDB-Entwickler noch das WebGenesis-Team zusätzliche Arbeit haben. Ein weiterer Vorteil ist, dass der FDB unabhängig von der WebGenesis-Version seine Anfragen stellen kann. Falls sich etwas an der Schnittstelle von WebGenesis ändert, muss nur der zusätzliche Web Service angepasst werden, ohne dass der FDB davon was mitbekommt.
Der Nachteil ist, dass der Web Service zusätzlich von einem Entwickler des Fachsystems gepflegt werden muss und dieser Entwickler Kenntnisse in WebGenesis und zum Teil vom FDB haben sollte.
Diese Lösung wird favorisiert, da hierbei weder zusätzlicher Aufwand auf der FDB-Seite noch auf Seiten von WebGenesis entsteht. Der Fachdokumentenbrowser kann die Anfragen an den Web Service stellen und muss dabei nur die Standards für SOAP-Anfragen beachten. WebGenesis braucht keine Anpassungen vornehmen, sondern muss nur die Schnittstellen beibehalten. Der Web Service zwischen den beiden Systemen ist flexibel gegenüber WebGenesis und dem Fachdokumentenbrowser, da er besser angepasst und erweitert werden kann als die beiden Systeme.

3.1.2. Lösungen für die Testkomponente

Die Testkomponente soll den Zugriff des zum Zeitpunkt dieser Diplomarbeit noch nicht implementierten Fachdokumentenbrowsers in Teilen prototypisch demonstrieren und simulieren. Die Funkti-

onalität der Komponente beschränkt sich auf das Anfordern aller Id's der Dokumente aus dem Shop der LUBW und das Ermitteln der Metadaten eines bestimmten Dokumentes. Damit gezeigt werden kann, dass die Daten auch korrekt erhalten wurden, werden sie einfach auf der Konsole oder einer anderen geeigneten Anzeigemöglichkeit (HTML) ausgegeben.

Die einfachste und schnellste Möglichkeit einen Web Service zu testen, ist der Aufrufe der Web Service Methoden über einen text- oder grafikbasierten Web Browser. Dazu muss nur die URL des Web Services und die Methodennamen und -parameter angegeben werden. Dies hat den Vorteil, dass der Test schnell und plattformunabhängig vollzogen werden kann, ohne Programmieraufwand zu haben. Der Nachteil ist hierbei, dass eine Mensch-zu-Applikation Kommunikation statt findet und dies nicht dem Gedanken von Web Services widerspiegelt. Des Weiteren können die erhaltenen Daten nicht programmtechnisch weiterverwendet werden.

Die andere Möglichkeit, um einen Web Service zu testen, ist die Verwendung eines Testclients, welcher in den verschiedensten Programmiersprachen implementiert werden kann. Die gewählte Programmiersprache muss nur das SOAP Protokoll verarbeiten können. Der Nachteil an dieser Lösung ist, dass ein Mehraufwand betrieben werden muss, um ein Ergebnis zu erhalten. Allerdings können bei dieser Lösung die Daten programmtechnisch weiter verarbeitet werden.

Um die Vorgaben für das Testen der Web Service Schnittstellen zu erfüllen, würde hier der Test mit Hilfe eines Web Browsers genügen. Hierbei wäre sichergestellt, dass der Web Service funktioniert und auch die geforderten Daten liefern kann.

Da zum Zeitpunkt dieser Diplomarbeit noch keine Testkomponente in Form einer Applikation existiert, welche den konzipierten und den implementierten Web Service mit dessen Schnittstellen testen kann, wurde zusätzlich ein Testclient entwickelt. Ergänzend kann bei ihm gezeigt werden, dass die Schnittstellen auch von anderen Programmiersprachen als Java optimal genutzt werden können. Für diese Zwecke wurde als Sprache Perl gewählt. Die geforderten Daten werden bei diesem Testclient auf der Konsole ausgegeben.

Im Gegensatz zu PHP braucht Perl keinen Webserver, um ein Programm ausführen zu können. Gegen die Verwendung von C# oder .NET spricht die zusätzlich Benutzung einer IDE[5] und die nicht Kompatibilität zu anderen Betriebssytemen. Perl hat den Vorteil, dass das Programm plattformunabhängig ist, der Code schnell implementiert werden kann und ein einfacher Editor genügt, um ein Programm zu entwickeln.

[5] Eine integrierte Entwicklungsumgebung (IDE) ist ein Anwendungsprogramm zur Entwicklung von Software.

3.2. Konzeptionelle Architektur der gewählten Lösungsstrategie

In den vorherigen Abschnitten wurden die verschiedenen Lösungsstrategien diskutiert und der Fokus auf die beste Strategie gelegt. Diese Strategie sieht einen extra implementierten Web Service vor, welcher die Schnittstellen für den Fachdokumentenbrowser bereitstellt.
Der Web Service soll ab sofort *WebShopWS*, kurz *WSWS*, heißen. Der *WSWS* ist eine eigenständige Komponente und damit nicht abhängig von WebGenesis oder dem FDB. Die Kommunikation vom FDB zum *WSWS* erfolgt ausschließlich über SOAP. Die Kommunikation vom *WSWS* zum WebGenesis-System kann wahlweise über SOAP oder über RMI erfolgen. Welches Kommunikationsprotokoll verwendet werden sollte, wird weiter unten genauer erklärt. Die Daten werden unidirektional vom Shop aus dem WebGenesis-System zum *WSWS* und vom Web Service zum FDB übertragen. Dies ist der Grund, weshalb der *WSWS* für den Login in WebGenesis nur einen Account besitzen muss, der Leserechte besitzt. Weiterhin sollte WebGenesis mindestens in der Version 7.10 vorliegen. In den früheren Versionen können Methoden fehlen, die für den *WSWS* wichtig sind.

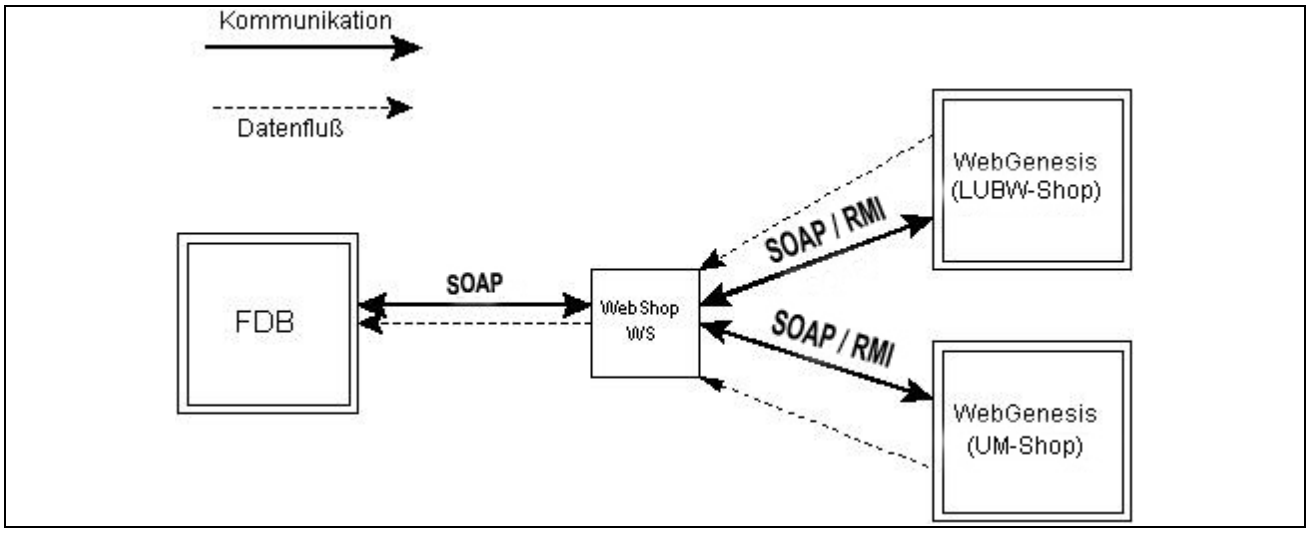

Abbildung 6: WebShopWS Kommunikationsaufbau

Wird der *WSWS* auf einem Server betrieben, auf dem nicht der LUBW-Shop liegt, erfolgt die Kommunikation durch die Firewall-Architektur der LUBW. Für diesen Kommunikationsweg ist SOAP als Protokoll mit Hilfe von HTTP die bessere Lösung zu RMI. RMI kann verwendet werden, wenn der *WSWS* auf dem gleichen Server liegt wie der Shop. Der Grund ist, dass nun die Kommunikation innerhalb eines Netzwerksegmentes erfolgt und hier keine Firewalls stören.

3.2.1. WebGenesis

Um in WebGenesis über die SOAP bzw. RMI Schnittstelle zu arbeiten, müssen lokal zwei Jar-Dateien (EDispatch.jar, jaxrpc.jar) aus WebGenesis importiert werden. Diese Jar-Dateien werden benutzt, um spezielle Methoden für die Ausführung der WebGenesis-Befehle zu erhalten. Die wichtigsten Methoden für die Ausführung sind die Methoden *execute* und *connect*. Die Methode *execute* verlangt als Parameter den Objektbezeichner (Entry, Global,…), dann eine Id des Eintrages, danach die eigentliche WebGenesis-Methode (get, search,…) und je nach Befehl einige Parameter, siehe Anhang Abbildungen zu WebGenesis-Befehle und in Abbildung 11, Zeile 4. Die *execute*-Methode gibt als Ergebnis ein Java Map Objekt zurück. Dieses Objekt wird im Code *result* genannt. Um die einzelnen Attribute des Map Objektes zu bekommen, z.B. den Namen des Eintrages oder die Id's der Untereinträge usw., wird mit der *get*-Methode (*result.get*) und den WebGenesis-Schlüsselbezeichner als Parameter der Wert ausgelesen, siehe Abbildung 11, Zeile 7. Bei Anfragen, welche mehr als einen WebGenesis-Befehl und Castings benötigen, wurden eigene Methoden implementiert (*getFileURI*, *getKeyWords*).

Die WebGenesis-Befehle können nicht nur über die SOAP bzw. RMI Schnittstelle aufgerufen werden, sondern auch über die URL per HTTP. In den Abbildungen im Anhang unter „WebGenesis-Befehle", steht „Remote" für SOAP bzw. RMI und „HTTP" für den Aufruf per URL. Alle Attribute welche unter „Parameter" stehen, verlangt der Befehl, um ordnungsgemäß ausgeführt werden zu können. Die „Rückgabewerte" sind alle diejenigen, welche *result.get* in die Java Map hinterlegt. Die Methode *connect* wird benutzt, um sich an das WebGenesis-System mit einem Benutzernamen und Passwort zu authentifizieren. Die genaue Arbeitsweise dieser Funktion kann im Entwicklerhandbuch von WebGenesis nachgelesen werden.

3.2.2. Schnittstellen des Web Services

Der Fachdokumentenbrowser fordert von den Web Services der Fachsysteme Schnittstellen an, welche ihm die Id's der Dokumente im Fachsystem und die Metadaten eines bestimmten Dokumentes liefern. Diese Schnittstellen müssen über SOAP erreichbar sein und die Daten in geeigneter Form zurück liefern. In geeigneter Form heißt soviel, dass die Rückgabewerte keinen programmiersprachenspezifischen Wertetyp haben, z.B. von Java eine „ArrayList".

Es gibt zwei Möglichkeiten, wie die Schnittstellen des Web Services aus der favorisierten Lösung (die dritte Lösung) benutzt werden können. Zum einen kann der Web Service eine Methode bereitstellen, die dem FDB erlaubt, alle Id's und die zugehörigen Metadaten der Dokumente mit einem Aufruf zu erhalten.

Zum anderen kann der Web Service getrennte Methoden anbieten, welche explizit eine Liste von Id's liefert. Diese Id's benutzt der FDB nun als Parameter für die zweite Methode des Web Services. Diese Methode liefert dem Fachdokumentenbrowser die Metadaten dieses speziellen Doku-

mentes. Der FDB muss also immer beide Methoden aufrufen, um die aktuellen Id's zu besitzen und die Metadaten eines Dokumentes zu bekommen.
Die Vorteile der ersten Variante sind, dass der FDB in der Lage ist, durch einen Aufruf alle aktuellen Dokumente zu erhalten. Der Nachteil ist, dass eine unnötige Netzlast erzeugt wird. Die Dokumente der Fachsysteme ändern sich selten und dann nur minimal.
Die zweite Variante hat den Vorteil, dass durch den ersten Methodenaufruf geprüft werden kann, ob neue Dokumente enthalten sind, was zugegeben auch bei der ersten Variante der Fall ist. Aber der FDB ist nun in der Lage, speziell die Metadaten der neuen Dokumente anzufordern und muss nicht die alten Daten wiederholt in seine Metadatenbank schreiben.
Es wird für den verwendeten Web Service die zweite Variante mit den zwei Methoden, welche einmal eine Liste von Id's und zum anderen die Metadaten eines bestimmten Dokumentes liefern, gewählt.

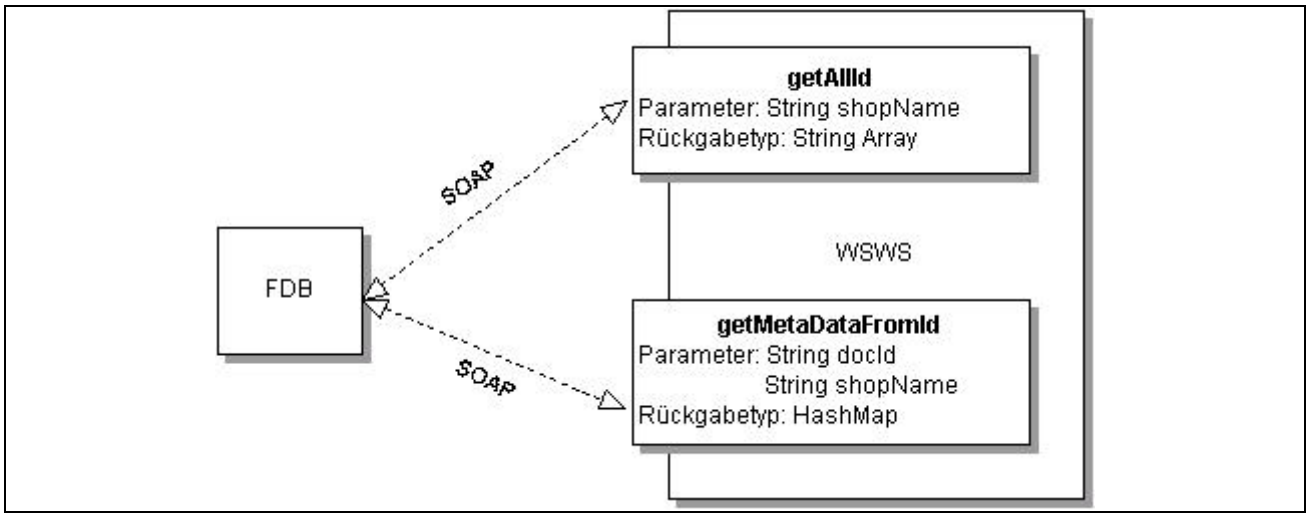

Abbildung 7: WSWS – Schnittstellen für FDB

Die erste Methode ist die *getAllId* Methode. Diese liefert dem FDB eine Liste, konkret ein String Array, zurück. In diesem Array sind alle Id's der Dokumente bzw. der Publikationen aus dem Shop enthalten. Als Parameter benötigt diese Methode nur den Namen des Shops, z.B. lubwShop. Dieser Parameter ist wichtig, damit die Methode weiß, welchen Shop aus WebGenesis, z.B. LUBW-Shop oder den UM-Shop, sie ansprechen soll. Das gleiche gilt bei der zweiten Methode. Die zweite Methode gibt eine HashMap zurück. In dieser HashMap sind alle Metadaten eines Dokumentes enthalten. Das zu bearbeitende Dokument wird durch die zu übergebende Id identifiziert.
Es wurde drauf geachtet, dass die Rückgabetypen keine speziellen Datentypen bzw. Objekte[6] sind. Dadurch können die Daten auch von anderen Programmiersprachen, sowohl OOP[7] (Java, C#), sowie nicht OOP Sprachen (Perl, PHP[8]) verarbeitet werden.

[6] mit Objekte sind hier spezielle Datenstrukturen in objektorientierten Programmiersprachen wie Java gemeint
[7] objektorientierte Programmierung
[8] ab Version 4.0 kann PHP auch OOP

Die Methode *getAllId* gibt ein String Array zurück, damit nur einmal alle Id's übertragen werden müssen und nicht jede Id einzeln. Die Methode *getMetaDataFromId* gibt eine HashMap zurück. Eine HashMap ist eine Java eigene Implementation eines Hashes[9]. Andere Programmiersprachen können diese HashMap wie einen normalen Hash verarbeiten. Hier wurde eine HashMap gewählt, da die Metadaten unterschiedliche Datentypen besitzen, siehe Tabelle 1. Darin liegt der Vorteil eines Hashes. Dieser Typ kann Daten mit den unterschiedlichsten Datentypen speichern. Eine andere Möglichkeit wäre hier die Verwendung einer Java ArrayList. Diese kann aber nicht von anderen Programmiersprachen verarbeitet werden und wurde deshalb verworfen.
Eine wichtige Aufgabe vom *WSWS* ist die Portierung der Datentypen der von WebGenesis gelieferten Metadaten in die geforderten Datentypen vom FDB. Es wurde mit den Verantwortlichen des FDB vereinbart, dass die Id's bzw. die Metadaten Strings sein sollen.

In der folgenden Tabelle werden der Mindestsatz an geforderten Metadaten, dessen WebGenesis-Datentypen und die portierten Datentypen durch den *WSWS* gegenübergestellt.

Metadaten	Bezeichnung in HashMap	WebGenesis-Datentyp	Portierter Datentyp
interner eindeutiger und persistenter Bezeichner (Id)	id	int	String, alle Id's zusammen als String Array
Titel	title	String	String
Kurzbeschreibung	kurzbeschreibung	String	String
thematische Zuordnung	thematischeZuordnung	nicht vorhanden	wird mit Hilfe der Properties-Datei erstellt
funktionale Klassifikation wie PortalU (Statusbericht, Rechtsvorschrift)	funktionaleKlassifikation	nicht vorhanden	wird mit Hilfe der Properties-Datei erstellt

[9] Hash ist ein Datentyp, welcher Werte mit unterschiedlichen Datentype speichern kann

Metadaten	Bezeichnung in HashMap	WebGenesis-Datentyp	Portierter Datentyp
Schlagworte	schlagworte	durch Id's verknüpft	Id's der Schlagworte müssen ermittelt werden und diese Schlagworte werden in String Array umgewandelt
Fachobjektbezug	fachobjektbezug	nicht vorhanden	Wird mit Hilfe der Properties-Datei erstellt
URL der Metadaten-Beschreibungsseite	urlMetaBeschreibungsseite	nicht vorhanden	wird zu einem String zusammengesetzt
Fachdokument	fachdokumentenDateien	Objekt mit allen Angaben (Name, Größe,…) zu jeder Datei im Dokument	FDB verlangt eine URI welche direkt die Datei öffnet. Alle URI's der Dateien sind in String Array gespeichert
Datum der letzten Änderung	datumLetzteAenderung	String	String

Tabelle 1: Übersicht über Datentypen der Metadaten

3.3. Probleme bei Änderungen

In der IT-Welt ist nichts von langer Dauer. Es wird immer kleine Änderungen geben. Damit der Web Service von außen angepasst werden kann, werden zusätzlich zum Programm noch Java Properties[10]-Dateien verwendet. Properties-Dateien sind Text-Dateien mit der Endung „.properties" in denen Schlüssel-Wert-Paare und Kommentare stehen. Diese Schlüssel-Wert-Paare sind in diesem Fall Verbindungsdaten, wie Benutzername und Passwort für das Einloggen in WebGenesis, sowie Webshop-spezifische Daten (Tabelle 2), welche durch spezielle Java Befehle ausgelesen und in Variablen gespeichert werden können.
Damit kann sichergestellt werden, dass ein Programm schnell angepasst werden kann, ohne im Code etwas zu ändern.

[10] Properties-Dateien sind Textdateien, die es ermöglichen, Werte für Attribute außerhalb des Programmes zu definieren

Schlüsselbezeichner	Wert	Bedeutung	Kommentar
$user	XXX	Benutzername damit *WSWS* sich authentifizieren kann	„$“ Zeichen für alle Nichtthemen. Wichtig für automatische Themenfindung.
$pwd	XXX	Passwort für die Authentifizierung in WebGenesis	
$serverUrl	www.lubw.baden-wuerttemberg.de	Shopserver URL	Hier kann auch anderer Server eingetragen werden, wo nach Einträgen gesucht werden kann
$serverPort	80/1099	Port für SOAP oder RMI Verbindung	
$communicationType	0/1	„0“ für SOAP „1“ für RMI	
$shopId	6638	WebGenesis-Shop Id	Falls Shop anderer Id bekommt
$entryCategory	19	Kategorie für Publikationen	Zusammen mit anderer Einstiegsid kann hier nach anderen Einträgen gesucht werden, z.B. „Termine“
Abfall	Abfall	Themenzuordnung. Schlüssel entspricht LUBW Thema, Wert entspricht PortalU Thema	
Chemikalien_und_Arbeitsschutz	Chemikalien; Gesundheit	Dito	LUBW Thema wird mehreren PortalU Themen zugeordnet. Getrennt durch „;“ Zeichen. „;“ Zeichen für spätere Verarbeitung wichtig, beliebige Zeichen möglich

Tabelle 2: Schlüssel-Wert-Paare aus Datei „lubwshop_service.properties“

3.4. Anpassen des Webshop Web Service

Die Webshop-spezifischen Daten sind Zuordnungen zwischen den verschiedenen Umweltkategorien bzw. Umweltthemen vom Shop der LUBW und dem Portal-U. Um die Themen später einzeln ansprechen zu können wurde das „;“ Zeichen als Trennzeichen gewählt. Weiterhin fällt das „$“

Zeichen bei einigen Schlüsselbezeichnern auf. Es wird für alle Nicht-Themen benutzt. Dadurch besteht die Möglichkeit, eine automatische Themenzuordnung zu erstellen und weitere Themenzuordnungen hinzuzufügen, ohne in den Quellcode des Web Services einzugreifen.
Damit der Web Service weiß, mit welchem WebGenesis-Shop er kommunizieren soll, wird ein Parameter beim Aufruf mitgegeben. Dieser Parameter gibt den Shop an (LUBW-Shop, UM-Shop etc.). Um nicht für jeden Shop einen extra Web Service zu schreiben, werden extra Properties-Dateien angeboten mit jeweils den Shopbezeichner im Namen, z.B. „lubwshop_service.properties“ oder „umshop_service.properties“. Die Inhalte der einzelnen Dateien unterscheiden sich minimal in den Verbindungsdaten und eventuell in den Umweltkategorien.
Damit der Web Service später auch auf andere WebGenesis-Eintragskategorien außer dem Shop mit dessen Publikationen angewendet werden kann, können die Eintragskategorie und das Einstiegsverzeichnis, für die Id und Metadatensuche, in der Properties-Datei geändert werden.

```
#++++++++++ConnectionProperties+++++++++#

$user=XXX

$pwd=XXX

$serverUrl=www.lubw.baden-wuerttemberg.de

$serverPort=80

# RMI(1)-> Port:1099 ist der Defaultwert von WebGenesis, SOAP(0)-> Port:80

$communicationType=0

#++++++++++ShopProperties+++++++++++++++#

#Id des WebShops in www.lubw.baden-wuerttemberg.de

$shopId=6638

$entryCategory=19

#++++++++++Themenassoziation++++++++++++++#

#WebShop Thema=PortalU Thema

Abfall=Abfall

Altlasten=Altlasten

…….

Lokale_Agenda_21=Verkehr;Umweltwirtschaft;Luft und Klima;Gesundheit;Energie

……
```

Abbildung 8: Properties-Datei“lubwshop_service.properties“ für WebShop Web Service

4. Realisierung

In den folgenden Kapiteln wird der programmtechnische Teil dieser Diplomarbeit näher vorgestellt. Es werden die wichtigsten Passagen der Methoden aus den Klassen des *WSWS* beschrieben. Es wird geklärt, warum die Web Service Methoden den Shop-Namen als Parameter fordern. Zum Schluss wird der Testclient und seine Implementierung näher erklärt.

4.1. Webshop Web Service (WSWS)

In den folgenden Abschnitten wird der *WSWS* genau erklärt, wie die Struktur des Web Services aufgebaut ist, was die wichtigsten Methoden sind und was sie machen. .

4.1.1. Entwicklungsumgebung für den Web Service

Als Entwicklungsumgebung wird das Open-Source-Framework Eclipse 3.1.2 IDE[11] benutzt. Für die Entwicklung von Web Services stellt das Lomboz[12] Projekt eine komplette Eclipse Version mit allen nötigen Tools bereit.
Diese Tools sind unter anderen Jasper, XDoclet, Axis, Ant und das Eclipse Java Development Toolkit (JDT). Zusätzlich empfiehlt sich einen Web Service fähigen Servlet Container, in diesem Fall Apache Tomcat 5.5, zu installieren. Eclipse ist in der Lage, den Servlet Container zum Testen des Web Services zu nutzen und kann mit Hilfe der Tools aus den Methoden und den Servereinstellungen die WSDL-Datei und die nötigen Konfigurationsdateien, z.B. die WSDD-Dateien, erstellen. Damit der Web Service entwickelt und getestet werden kann, muss eine Java Runtime Environment auf dem Entwicklungssystem installiert sein. Für dieses Projekt wird die Java Runtime Environment 1.5 verwendet.

4.1.2. Klassenstruktur vom WSWS

Der *WSWS* besteht aus drei Klassen. Die Klassen mit den Bezeichnungen „Fachdokument“ und „FileHandler“, sowie die Klasse Namens „WebShopWS“. Die Klasse „FileHandler„ ist eine Hilfsklasse, welche eine Methode zum Finden der Properties-Datei bereitstellt. Die Klasse „Fachdokument“ hat nur die Aufgabe die Attribute (Titel, Id, letzte Änderung usw.) eines Fachdokumentes bereitzustellen und sie gebündelt der Klasse „WebShopWS“ zu übergeben. Das Setzen der einzelnen Attribute des Fachdokumentenobjektes, sowie das Bereitstellen der Schnittstellen für die Kommunikation mit dem Fachdokumentenbrowser und dem CMS, übernimmt die

[11] Eine integrierte Entwicklungsumgebung (IDE) ist ein Anwendungsprogramm zur Entwicklung von Software.
[12] ObjectWeb Lomboz for JavaEE Development ist ein Open-Source-Projekt. Es vereinfacht die Entwicklung von Web Services in Eclipse. Lomboz ist unter der Url http://www.objectlearn.com/index.jsp

„WebShopWS“ Klasse. Damit der Fachdokumentenbrowser die beiden angesprochenen Methoden erreichen kann, sind diese als „public“ deklariert. Alle anderen, außer dem Konstruktor sind „private“ und können nicht von außen erreicht werden.

Das folgende UML Diagramm zeigt sämtliche Attribute und Methoden der beiden Klassen. Dabei sind die als „public“ deklarierten Attribute und Methoden mit einem „+“ und die als „private“ deklarierten Attributen und Methoden mit einem „-“ Zeichen markiert. Die wichtigsten Methoden werden im Abschnitt „Methoden des *WSWS*“ vorgestellt.

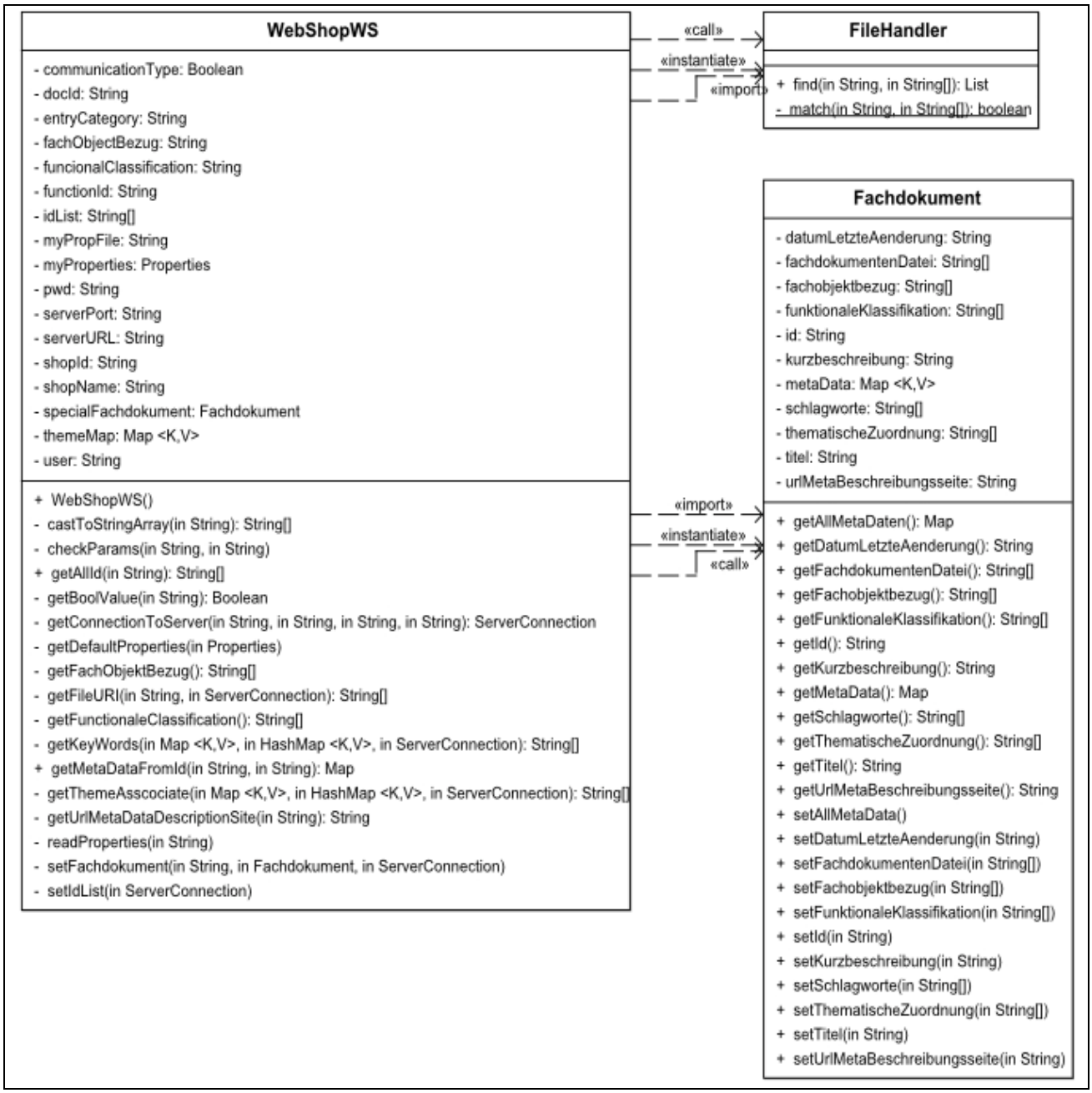

Abbildung 9: WebShopWS UML Diagramm

4.1.3. Methoden des *WSWS*

Nun werden die wichtigsten Methoden der Klasse „WebShopWS“ vorgestellt. Diese Methoden sind *readProperties*, *setFachdokument*, *getFileURI , getThemeAsscociate*, *setIdList*, und zum Schluss die Web Service Methoden *getMetaDataFromId* und *getAllId*.

Die Methode *readProperties* wird von den Web Service Methoden *getMetaDataFromId* und *getAllId* aufgerufen, Sie liest alle Schlüssel-Wert-Paare aus einer Properties-Datei aus und speichert die Werte in Variablen (ab Zeile 12).
Damit der *WSWS* unabhängig von der Verzeichnisstruktur des Servers ist, sucht die Methode *readProperties*, mit Hilfe der Methode *find* aus der Klasse *FileHandler*, ab dem Arbeitsverzeichnis des eingesetzten Tomcat-Servers rekursiv nach der Properties-Datei. Dazu ermittelt er den Einstiegspunkt des Arbeitsverzeichnisses (Zeile 2) und speichert alle Dateien mit der Endung „.properties“ in eine Liste von File-Objekten (Zeile 4). Diese durchläuft er mit Hilfe einer Schleife (Zeile 5) und bricht die Suche ab, wenn die richtige Properties-Datei gefunden wurde. Daraufhin wird der absolute Pfad in die Variable *myPropFile* speichert. Hier kommt der Parameter beim Aufruf der Web Service Methoden ins Spiel. Die Methoden verlangen unter anderem einen Parameter für den Shop-Namen. Jeder Shop besitzt eine eigene Properties-Datei. Diese Properties-Dateien beginnen mit dem Shop-Namen und enden mit „_service.properties“. Durch den Parameter beim Aufruf findet die Methode *readProperties* die richtige Properties-Datei. Die Variable *myPropFile* wird in der Zeile 12 verwendet, um die Werte aus der Properties-Datei zu lesen.
Ab der Zeile 17 werden die Umweltkategorien automatisch aus der Properties-Datei gesucht und gespeichert. Hier wurde auf eine statische Themenzuordnung verzichtet, da sich die Umweltthemenzuordnung ändern kann. Es werden alle Schlüsselbezeichner in eine „Enumeration“ gespeichert (Zeile 17) und diejenigen die kein „$“ Zeichen zu Beginn haben, als Umweltkategorie gespeichert (ab Zeile 20).

```
1   private void readProperties(String functionId) {
2     String path = new File(System.getProperty("user.dir")).getAbsolutePath();
3     FileHandler fh = new FileHandler();
4      List<File> files = fh.find( path,  ".properties" );
5      for ( File f : files ){
6          if (f.getName().equalsIgnoreCase(this.shopName.toLowerCase() + "_"+ "service.properties")) {
7                                   this.myPropFile = f.getAbsolutePath();
8                                   break;
9          }
10     }
11         try {
12                  prop.load(new FileInputStream(this.myPropFile));
13                  this.user = prop.getProperty("$user");
14                  this.pwd = prop.getProperty("$pwd");
15                  ……
16                  // findet automatisch die Zuordnungen der Themen aus Properties - Datei
17                  Enumeration e = prop.propertyNames();
18                  String[] propertiesElements = new String[prop.size()];
19                  int i = 0;
20                  while (e.hasMoreElements()) {
21                              propertiesElements[i] = (e.nextElement()).toString();
22                              if (!(propertiesElements[i].startsWith("$"))) {
23                                          i++;
24                              }
25                  }
26                  for (int j = 0; j < propertiesElements.length; j++) {
27                              if (!(propertiesElements[j] == null)) {
28                                          this.themeMap.put(propertiesElements[j], prop.get(propertiesElements[j]));
29                              }} …..}
```

Abbildung 10: WebShopWS Klasse – readProperties

Die folgende vorgestellte Methode der Klasse „WebShopWS“ heißt *setFachdokument* und ist für das Setzen der Attribute des Fachdokumentenobjektes, welche die Metadaten eines Fachdokumentes repräsentieren, verantwortlich. In der Zeile 4 werden die Informationen aus einem WebGenesis-Entry-Objektes wie unter Abschnitt 4.2.1 erläutert herausgeholt und in *result* gespeichert.

```
1  private void setFachdokument(String docId, Fachdokument fd, ServerConnection con) {
2                  …….
3                  HashMap params = new HashMap();
4                  Map result = con.execute("entry", docId, "get", params);
5                  // set metadata
6                  fd.setId(docId);
7                  fd.setTitel(result.get("title").toString());
8                  fd.setKurzbeschreibung(result.get("Abstract").toString());
9                  fd.setFachdokumentenDatei(getFileURI(docId, con));
10                 fd.setSchlagworte(getKeyWords(result, params, con));
11                 ….....
12 }
```

Abbildung 11: WebShopWS Klasse – setFachdokument

Da der Fachdokumentenbrowser eine vollständige URI der Datei verlangt, muss sie durch die Methode *getFileURI* generiert werden. Mit dieser URI kann die Datei direkt aufgerufen werden. Zeile 3 führt wieder einen WebGenesis-Befehl aus, um die Informationen aller Dateien eines bestimmten Eintrages zu erhalten. Durch Zeile 4 werden alle Dateinamen der Dateien des Eintrages in ein Array gespeichert. Mit Hilfe der Größe des Arrays „dateiListe“ wird ein Hilfs-Array initialisiert und in dieses Hilfs-Array die einzelnen URI’s gespeichert. Da WebGenesis einen speziellen Aufbau der URI besitzt, um eine Datei direkt aufrufen zu können, muss diese explizit für jede Datei zusammengesetzt werden (Zeile 9).

```
1 private String[] getFileURI(String docId, ServerConnection con) throws RemoteException {
2         HashMap params = new HashMap();
3         Map result = con.execute("entry", docId, "getContentFileInfos", params);
4         String[] dateiListe = (String[]) (result.get("name"));
5         ……
6         String[] dateiListenArray = new String[dateiListe.length];
7         ……
8         for (int i = 1; i < dateiListe.length; i++) {
9                 newDateiURI = "http://" + serverURL + "/servlet/is/" + docId + "/" + dateiListe[i].toString()
                          + "?command=downloadContent&filename=" + dateiListe[i].toString();
10                dateiListenArray[i] = newDateiURI;
11            }
12        ……
13 }
```

Abbildung 12: WebShopWS Klasse – getFileURI

Das Verarbeiten der Zuordnungen aus der Properties-Datei übernimmt die Methode *getThemeAsscociate*. In der Zeile 3 wird die Id des Elternverzeichnisses ermittelt. Dieses Verzeichnis entspricht der Umweltkategorie. Durch diese Id wird der Titel ermittelt und in die Variable *shopCategory* gespeichert (Zeile 4). Mit Hilfe des Titels kann nun die Zuordnung, wie sie in der Properties-Datei erstellt worden ist, ermittelt werden (Zeile 6).

Hinter der Zeile 6 steckt folgendes:

- shopCategory.replace(" ", "_") → Hier wird aus z.B. „Allgemeine Umweltfragen“ folgender String „Allgemeine_Umweltfragen“. Dies ist nötig, damit die Kategoriebezeichner die gleichen sind wie in der Properties-Datei.
- this.themeMap.get(…) → Nimmt den Kategoriebezeichner z.B. „Allgemeine_Umweltfragen“ als Schlüsselbezeichner und liefert die Werte aus der Properties-Datei zurück
- castToStringArray(...) → Da die Werte aus der Properties-Datei als ein String übergeben werden, muss dieser noch getrennt werden. Um dies zu realisieren wurde, wie unter Abschnitt 3.4 geschildert, das Zeichen „;“ als Trennzeichen benutzt.

```
1   private String[] getThemeAsscociate(Map result, HashMap params, ServerConnection con) {
2                       ……
3                       String parentID = ((Integer) (result.get("parent"))).toString();
4                       result = con.execute("entry", parentID, "get", params);
5                       String shopCategory = (result.get("title")).toString();
6                       return castToStringArray(this.themeMap.get(shopCategory.replace(" ", "_")).toString());
7                       …….
8   }
```

Abbildung 13: WebShopWS Klasse – getThemeAssociate

Die Methode *setIdList* macht die eigentliche Arbeit von *getAllId*. Sie unterscheidet sich von den anderen vorgestellten Methode in dem Sinne, dass die Variable *params* für die WebGenesis-Anfrage *con.execute("entry", "", "search", params*“) mit Werten belegt werden muss.
Die WebGenesis-Anfrage ist eine Suche nach Einträgen mit einer bestimmten Kategorie (Zeile 4), welche unterhalb des Webshops bzw. eines anderen Einstiegsverzeichnisses gesucht werden (Zeile 5). Damit in den Untereinträgen, sprich rekursiv, gesucht wird, ist der Parameter „use-SubTree“ auf „true“ gesetzt. Als Ergebnis dieser Suche wird ein Integer Array mit den Id’s der Publikationen geliefert, welche aufgrund der Anforderungen noch in ein String Array gecastet werden müssen (Zeile 10 bis 14). Das fertige String Array ist eine globale Variable und kann später direkt von *getAllId* aufgerufen werden.

```
1       private void setIdList(ServerConnection con) {
2               ……
3               HashMap params = new HashMap();
4               params.put("categories", this.entryCategory);
5               params.put("subTree", Integer.valueOf(shopId));
6               params.put("useSubTree", Boolean.valueOf(true));
7               Map result;
8               result = con.execute("entry", "", "search", params);
9               int[] myIdList = (int[]) (result.get("resultsSE"));
10              this.idList = new String[myIdList.length];
11              if (myIdList != null) {
12                      for (int i = 0; i < myIdList.length; i++) {
13                              this.idList[i] = (Integer.valueOf(myIdList[i])).toString();
14                      }
15              ……
16  }
```

Abbildung 14: WebShopWS Klasse – setIdList

Die Methode *getAllId* ist eine von den zwei geforderten Schnittstellen des Web Services. Sie liefert die Id's der Publikationen aus dem Webshop. Dafür verbindet sie sich zuerst mit dem WebGenesis-Server (Zeile 3) mit Hilfe der selbst implementierten Methode *getConnectionToServer*, welche die EDispatcher-Methode *connect* aufruft. Mit verbinden ist hier gemeint, dass eine kurze Verbindung zu dem WebGenesis-Server hergestellt wird, damit sich der *WSWS* authentifizieren kann. Bei erfolgreicher Authentifizierung liefert WebGenesis ein Objekt, das *Connection-Objekt,* (Zeile 3) zurück, welches an die Methode *setIdList* übergeben wird (Zeile 4). Nun kann *setIdList* mit diesem Objekt arbeiten, sie speichert, wie oben erklärt, eine Liste mit den Id's der Dokumente aus dem Shop der LUBW in eine globale Variable. Diese Variable bzw. die Liste der Id's wird nun an den FDB oder einen anderen Client zurückgeliefert (Zeile 6).

Aus folgenden Gründen wurde hier als Rückgabewert ein String Array gewählt:

1. Eine Id muss nicht zwingend eine Zahl, sondern kann auch eine Folge von Zeichen sein.
2. Ein Array ist besser zu verarbeiten, als eine große Zeichenkette.

```
1   public String[] getAllId(String _shopName) {
2           …….
3           ServerConnection con = getConnectionToServer(serverURL, serverPort, user, pwd);
4           setIdList(con);
5           con.disconnect();
6           return idList;
7           ……
8   }
```

Abbildung 15: WebShopWS Klasse – getAllId

Die andere Methode, die durch einen Client über den Web Service genutzt werden kann, ist *getMetaDataFromId*. Sie liefert die Metadaten eines Dokumentes zurück. Dazu muss die Methode eine Verbindung zum WebGenesis-Server erstellen. Mit Hilfe des *Connection-Objektes*, der Id des Dokumentes und einem Fachdokumenten-Objekt, kann die Hilfs-Methode *setFachdokument* (Zeile 4) die Metadaten ermitteln und setzen. Das neue Fachdokumenten-Objekt repräsentiert das Dokument aus dem Shop mit allen Metadaten. Diese Metadaten werden durch den Aufruf der Funktion *setAllMetaData* in eine HashMap gepackt (Zeile 6) und gesammelt als Rückgabewert zurückgeliefert (Zeile 7).

```
1   public Map getMetaDataFromId(String _docId, String _shopName) {
2           ……
3           ServerConnection con = getConnectionToServer(serverURL, serverPort, user, pwd);
4           setFachdokument(this.docId, specialFachdokument, con);
5           con.disconnect();
6           specialFachdokument.setAllMetaData();
7           return specialFachdokument.getAllMetaDaten();
8           ……
9   }
```

Abbildung 16: WebShopWS Klasse – getMetaDataFromId

Dazu initialisiert die Methode eine globale HashMap namens „metaData“. In dieser HashMap werden die einzelnen Attributwerte mit einem aussagekräftigen Schlüsselbezeichner gespeichert.

Die HashMap kann mit der dazugehörigen Methode (getAllMetaData) von der „WebShopWS“ Klasse geholt werden.

```
public void setAllMetaData() {
        this.metaData = new HashMap();
        this.metaData.put("id", getId());
        this.metaData.put("titel", getTitel());
        this.metaData.put("kurzbeschreibung", getKurzbeschreibung());
        ….
}
```

Abbildung 17: Fachdokument Klasse – setAllMetaData

4.1.4. Einrichten des *WSWS* unter WebGenesis

WebGenesis besitzt, wie in Abschnitt 2.4.2 erwähnt, eigene Web Services. Diese Web Services kann man unter der URI http://SERVER/servlet/is/services/WebGenesis? aufrufen. Damit der *WSWS* unter einer ähnlichen URI aufgerufen werden kann, wird er unter WebGenesis eingebunden. Für den ersten Test wurde der *WSWS* auf dem Testserver http://lfu.server.de eingerichtet.

Folgende Schritte müssen ausgeführt werden, um den Web Service in die WebGenesis-Systemumgebung einzubinden:

1. WebShopWS.jar, welche die Klassen des *WSWS* enthält, muss nach
   app/base/webapps/WebGenesis/WEB-INF/lib
   kopiert werden.

2. In:
   app/base/webapps/WebGenesis/WEB-INF/server-config.wsdd
   muss der Inhalt von
   Abbildung 17
   nach den Services von WebGenesis eingebaut werden:

```
<service name="Version" provider="java:RPC">
      <parameter name="allowedMethods" value="getVersion" />
      <parameter name="className" value="org.apache.axis.Version" />
    </service>

    <service name="WebShopWS" provider="java:RPC" style="wrapped" use="literal">
       ...
 </service>
```

3. Die Datei

   lubwshop_service.properties bzw. die Properties-Dateien der anderen Shops muss nach

   app/base/conf/

   kopiert werden.

4. Der WebGenesis-Server muss neu gestartet werden:

   cd /home/lfu/app/admin

   control restart

5. Über die Auflistung der Services mit:

   http://lfu.server.de/servlet/is/services#/

   kann die WSDL eingesehen werden.

6. Der Dienst kann aufgerufen werden mit:

   http://lfu.server.de/servlet/is/services/WebShopWS?method=getAllId&shop=lubwshop

   oder

   http://lfu.server.de/servlet/is/services/WebShopWS?method=getMetaDataFromId&id=4711&shop=lubwshop

Die Einbindung vom *WSWS* in andere Systeme kann sich unterscheiden. Der *WSWS* kann in jedes System integriert werden, welches Tomcat als Webserver benutzt. Es ist sogar möglich den *WSWS* als Standalone Lösung laufen zu lassen, d.h. dass es ist kein WebGenesis, sondern nur Tomcat benötig. Dies kann durch Tools, z.B. Eclipse mit Lomboz und den *WSWS* Klassen erreicht werden. In dieser Diplomarbeit wird nicht näher auf die Standalone Lösung eingegangen.

4.2. Testclient

Der Testclient ist in Perl implementiert sieh Abschnitt 3.1.2. Perl bietet die Möglichkeit auf verschieden Wege zum Ziel zu kommen, getreu dem Perl Motto „here is more than one way to do it“.

Dadurch entsteht ein sehr großer Freiheitsgrad und das Programm kann an die Systemumgebung und die speziellen Anforderungen anpassen werden. Für Perl gibt es viele zusätzliche Module, um die Fähigkeiten der Programmiersprache zu erweitern. Wegen der Vielfalt dieser Module wurde am 25. Oktober 1995 das CPAN (Comprehensive Perl Archive Network) Projekt ins Leben gerufen. CPAN ist ein weltweites gespiegeltes Online-Repository für Perl-Module. Heute enthält CPAN über 8.200 Module von über 4.400 Autoren. Um einen Web Service Client bzw. einen Web Service Server in Perl zu implementieren, empfiehlt sich das Module „SOAP::Lite“ zu verwenden. Dieses Modul kann über CPAN erhalten und in den Code eingebunden werden. Mit Hilfe von „SOAP::Lite“ ist Perl in der Lage, SOAP Nachrichten zu erstellen, zu empfangen, zu verarbeiten und welche zu verschicken.

Im nächsten Abschnitt wird der Web Service Client Code vorgestellt und die wichtigsten Passagen genauer erklärt. Der Web Service Client kann auf einem beliebigen PC liegen. Dieser PC muss nur eine funktionierende Internetverbindung und Perl mit den benötigten Modulen (SOAP::Lite) haben.

4.2.1. Module für Perl Client

Damit der implementierte Perl Client fehlerfrei funktioniert, müssen einige Perl Module auf dem Clientrechner installiert sein. Für diese Diplomarbeit wurden die Module von www.cpan.org herunter geladen und unter der Cygwin[13]-Umgebung installiert.

Verwendete Module:

- expat-2.0.0
- SOAP-Lite-0.66
- XML-Parser-2.34
- Compress-Zlib-1.41
- HTML-Parser-3.54
- HTML-Tagset-3.10
- Libwww-perl-0.02
- MIME-Base64-3.07
- WebService-0.01

4.2.2. Web Service Client Code

Damit das Programm SOAP Anfragen an den Web Service schicken und die Antworten auch verarbeiten kann, wurde hier das Modul „SOAP::Lite“ verwendet (Zeile 2). Der Client muss als erstes

[13] Cygwin ist eine Sammlung freier Software, die unter verschiedenen Versionen von Microsoft Windows eine Vielzahl von Funktionen einer UNIX-Umgebung bereitstellt.

eine Verbindung zum Web Service, welcher hier unter der URI „http://lfu.server.de/servlet/is/services/WebShopWS“ (Zeile 5), herstellen. Jetzt besteht die Möglichkeit, über einen Proxy, die vom Web Service angebotenen Methoden anzusprechen oder es wird die WSDL-Datei des Web Services verwendet.

Die Methoden werden mit einem einfachen Referenzaufruf angesprochen und die Rückgabewerte weiter verarbeitet, wie in den Zeilen 8 und 9 zu sehen ist. Der weitere Verarbeitungsaufwand hängt jetzt vom Rückgabewert und von den Möglichkeiten von Perl ab. Der unterschiedliche Verarbeitungsaufwandes ist an diesem Code recht gut zu sehen. Die Verarbeitung des String Arrays (Zeile 8, 9) ist sehr einfach, wohingegen die Verarbeitung des Hashes ab der Zeile 15 weit aus komplizierter ist.
Hier müssen zuerst die vorhandenen Schlüsselbezeichner herausgefunden werden und dann die zugehörigen Werte. Da der vom Web Service zurückgelieferte Hash nicht nur Strings, sondern auch String Arrays liefert, müssen auch diese Schlüssel-Wert-Paare ausgelesen werden. Diese Problematik wird in den Zeilen 15 bis 25 gezeigt. Hier schaut das Programm, ob es ein String Array oder nur ein String ist und entscheidet dann, wie es fortfahren soll (Zeile 18). Wenn es ein String ist, dann liefert er die Schlüsselbezeichnung und den zugehörigen Wert (Zeile 19). Ist es ein String Array, muss das Programm zuerst die Schlüsselbezeichnung und den zugehörigen Wert des Hashes ermitteln (Zeile 21). Es wird der alte Wert des Schlüssel-Wert-Paares als Schlüsselbezeichner verwendet und der zugehörige Wert ausgegeben (Zeile 23). Dieser Wert sind die einzelnen Strings des String Arrays.

```
1   …..
2   use SOAP::Lite;
3   …..
4   my $soap = SOAP::Lite
5           ->uri(' http://lfu.server.de/servlet/is/services/WebShopWS ')
6           ->service (' http:// lfu.server.de/servlet/is/services/WebShopWS?wsdl');
7   ….
8   my @idList = $soap->getAllId($wgDienst);
9   foreach my $id (sort (@idList)){  printf "Id: %s ||",$id; }
10  ……..
11  my $hash = new $soap->getMetaDataFromId($docId);
12  …….
13  #+++++ gibt Schlüssel-Wert-Paare auf Konsole aus ++++#
14  ……
15  foreach $k (sort(keys %{$hash}) )
16  {
17          if ($hash->{$k}){
18                  if (!($hash->{$k} =~ /^HASH/)){
19                  printf "key: %s\nvalue: %s\n", $k,$hash->{$k};
20                  }else{
21                          foreach $j ( sort(keys %{$hash->{$k}}) ) {
22                                  printf "key: %s\n", $k;
23                                  printf "value: %s\n\n", $hash->{$k}{$j};
24                  }
25          }
26          …….
27  }
```

Abbildung 18: Web Service Client Code

5. Schlusswort

In dieser Diplomarbeit wurde ein Feinkonzept für die Integration der Fachsysteme im UIS mit dem Fachdokumentenbrowser (FDB) des FZK über Web Services erstellt. Es wurde dieses Feinkonzept speziell an den Shop der LUBW angepasst. Es beinhaltet die Implementierung eines Web Services, welcher zwei Schnittstellen für den FDB bereitstellt. Diese Schnittstellen müssen dem FDB die Id's aller Dokumente, im Fall der LUBW die der Publikationen des Webshops, und die Metadaten eines bestimmten Dokumentes liefern. Die Funktionalität der Schnittstellen sollte abschließend getestet werden.

Es wurden im Rahmen der Diplomarbeit mehrere Lösungswege für die Verwendung eines Web Services diskutiert. Diese Lösungsstrategien beinhalteten die Verwendung der WebGenesis-eigenen Web Service Schnittstelle bis hin zu der Implementierung eines eigenen Web Services. Es wurde für das Feinkonzept die Verwendung eines eigenen Web Services als beste Lösung gesehen. Weiterhin wurden die Schnittstellen implementiert und erklärt. Um die Schnittstellen testen zu können, wurden mehrere Möglichkeiten für einen Test gegenübergestellt. Für die Erfüllung der eigentlichen Aufgabe, nämlich das Testen der Funktionen der Schnittstellen, hätte der Aufruf des Web Services über einen Web Browsers ausgereicht.
Um zusätzlich beweisen zu können, dass der Web Service auch programmiersprachenunabhängig von einer Applikation angesprochen werden kann, wurde für diese Zwecke ein Testclient entwickelt. Es wurden die verschiedenen Programmiersprachen diskutiert und als beste Lösung Perl gewählt. Weiterhin sind der Code und die verwendeten Module erklärt worden. Es wurden alle Vorgaben und Anforderungen Seitens der LUBW und dem FDB eingehalten und ergänzend gezeigt, dass der Web Service durch eine Applikation, unabhängig der implementierten Sprache, angesprochen werden kann.

Für die Erstellung des Feinkonzeptes mussten die beteiligten Systeme in Ihren Funktionalitäten und Fähigkeiten untersucht werden. Da der Fachdokumentenbrowser zum Zeitpunkt der Diplomarbeit nur im Grobkonzept existiert, konnten keine Dokumentationen zum FDB herangezogen werden. Das WebGenesis-System ist schon länger in der Produktion vorhanden und daher gibt es auch geeignete Dokumentationen, die jedoch durch Informationen direkt von dem Entwickler ergänzt werden musste.

Zukünftige Weiterentwicklungen am Web Service könnten generische Methoden beinhalten, die eine Vielzahl von Anforderungen bewältigen können. Ein Beispiel könnte das Finden von Dokumenten sein, welche in einem bestimmten Zeitraum erstellt worden sind. Dadurch wäre der Web Service noch flexibler.

Mit der Diplomarbeit wurde erfolgreich nachgewiesen, dass das Grobkonzept zum künftigen UIS-Fachdokumentenmanagement sinnvoll verfeinert und umgesetzt werden kann. Damit wurde das Ziel der Aufgabe erreicht. Im Laufe der Arbeit wurde die Web Service Schnittstelle von WebGenesis zum ersten Mal in der LUBW genutzt. Damit wurden wertvolle Erfahrungen gesammelt, die auch für künftige Anwendungen in der LUBW hilfreich sind.

V. Literatur

R. Weidemann, R. Ebel, R. Mayer-Föll : Fachdokumentenmanagement im Umweltinformationssystem Baden-Württemberg. Forschungszentrum Karlsruhe, Karlsruhe 2005

J. Moßgraber, F. Chaves, F. Kaiser, U. Bügel, F. Walter: Entwicklungshandbuch für WebGenesis Version 7.10. Rev. 3.2, Fraunhofer Institut Informations- und Datenverarbeitung, Karlsruhe 2005

Literatur aus dem Internet

- Building a simple Web Service
  http://www.roseindia.net/webservices/buildingsimplewebservice.shtml
- SOAP::Lite
  http://cpan.uwinnipeg.ca/htdocs/SOAP-Lite/SOAP/Lite.html
  http://soaplite.com/
- CPAN
  http://cpan.org/
- Web Service
  http://de.wikipedia.org/wiki/Web_Service
- Lomboz
  http://www.objectlearn.com/index.jsp
- Omondo
  http://www.omondo.de/

VI. Quellen

- Abbildung 1: Grobkonzept des Aufbaus vom Fachdokumentenbrowser
  - Studie „Fachdokumentenmanagement im Umweltinformationssystem Baden-Württemberg“
- Abbildung 4: Web Service Roles
  - http://de.wikipedia.org/wiki/Web_Service
- Abbildung 5: Aufbau einer SOAP Nachricht
  - http://www.tecchannel.de/sicherheit/grundlagen/402212/index4.html
- Abbildung 19, 20, 21: Befehl zum Objekttype Entry
  - Entwicklerhandbuch WebGenesis