TECHNISCHE UNIVERSITÄT DRESDEN

Klassifizierung und Suche Semantischer Ressourcen in verteilten Systemen

Andreas Hollmann

30.11.2008

Verantwortlicher Hochschullehrer:

Prof. Dr. rer. nat. habil. Uwe Aßmann

Betreuer:

Dipl.-Ing. (FH) Gerald Faschingbauer

Institut für Bauinformatik, Prof. Dr.-Ing. Raimar J. Scherer

Dipl.-Inf. Sebastian Richly

Lehrstuhl Softwaretechnologie, Prof. Dr. rer. nat. habil. Uwe Aßmann

Eigenständigkeitserklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.
Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Dresden, 30 November 2008

Andreas Hollmann

Vorwort

Ich möchte Herrn Professor Aßmann für die Vergabe des Themas dieser Diplomarbeit danken und für das Grundwissen, das er uns Studenten während seiner Vorlesungen vermittelt hat. Ebenso möchte ich mich bei Gerald Faschingbauer bedanken. Von ihm habe ich nicht nur viele fachliche Dinge gelernt, sondern auch vieles, was mich sicherlich besser auf die Welt außerhalb des Studiums vorbereitet. Ebenso möchte ich mich bei Sebastian Richly bedanken für die Unterstützung und Betreuung während der Integration der Konzepte aus dieser Arbeit in die OSPP. Außerdem möchte ich mich bei meinem guten Freund und Kommilitonen Alexander Wülfing für wertvolle Diskussionen und Tipps bedanken und ebenso für die gemeinsam verbrachte Zeit. Alexander Gehre möchte ich meinen herzlichsten Dank aussprechen für die Hilfe beim Verstehen komplexer semantischer Technologien und Ulf Wagner für die Bereitstellung der benötigten technischen Infrastruktur zur Implementierung und Evaluierung der während dieser Arbeit erstellten verteilten Anwendung. Jördis Wild möchte ich für ihre wertvolle baufachliche Zuarbeit zum Evaluierungsbeispiel danken.

Ich möchte ebenso meinen Eltern danken. Sie haben von Allen den größten Teil zu dieser Arbeit beigetragen, denn sie haben mich erzogen und zu dem Menschen gemacht, der ich bin.

Aufgabenstellung

Web Services zeichnen sich immer mehr als geeignete und zielführende Technologie für die Entwicklung Serviceorientierter Architekturen (SOA) aus. Eine SOA – Anwendung ist nicht mehr monolithisch aufgebaut, sondern sie setzt sich aus mehreren Komponenten zusammen. Die Web Services können in beliebiger Sprache programmiert, auf verschiedenen Plattformen ausgeführt und von verschiedenen Anbietern als Onlinedienst angeboten werden. Diese Technologie wird es künftig ermöglichen, flexibel anpassbare Informationssysteme für die Anwendung in der Praxis zu entwickeln. Das Institut für Bauinformatik arbeitet an der Entwicklung einer IT-Infrastruktur für die Überwachung von Bauwerken während der Bauausführung. Durch den Einsatz verteilter Ressourcen sollen die Möglichkeiten zur Nachrechnung und Aktualisierung von geotechnischen Entwürfen sowie die Verwaltung von Mess-, Versuchs- und Prognosedaten bei baubegleitenden Überwachung geotechnischer Ingenieurbaumaßnahmen entscheidend verbessert werden.
Ziel der Diplomarbeit ist es, die am Institut für Software- und Multimediatechnik entwickelte OSPP-Open Web Service Plattform um die semantische Beschreibung der eingebundenen Ressourcen zu ergänzen und sie zur Bearbeitung der Problemstellungen der Bauwerküberwachung anzupassen.
Zielsetzung und angestrebte Arbeitsergebnisse:

- Erarbeitung einer Methode zur facettenbasierten Klassifizierung von Software Komponenten (Web Services)
- Semantische Suche (es existieren mehrere semantische Such-Algorithmen, manche sind noch nicht implementiert) für in OWL-S beschriebene Ressourcen (Webservices und mechanische Modelle aus dem Bauwesen)
- Integration von in OWL-S beschriebenen Ressourcen in einen Workflow
- Anpassung der OSPP-Open Web Service Plattform an die Ingenieuraufgabe
- Prototypische Implementierung von Domänenontologien und Webservices aus dem Bauwesen
- Ausführung eines Evaluierungbeispiels

Inhaltsverzeichnis

Einleitung

Semantic Web ist eines der meistgenutzten Schlagworte im Bezug auf die Zukunft des WWW. Die Notwenigkeit des Semantic Web entsteht durch die „Überproduktion“ von Information. Dabei werden mehr Informationen produziert, als verarbeitet werden können. Dabei verlieren die produzierten Informationen ihre Nützlichkeit, denn die Information hat nur dann einen Nutzen, wenn diese verarbeitet und in der Schlussfolgerung eines Ergebnisses berücksichtigt werden kann. Die obere Aussage wird durch die folgende Statistik belegt:

Die Anzahl der im Internet verfügbaren Webseiten und die Menge der angebotenen Services wächst stetig an. Bereits im November 2006 ergaben Schätzungen, dass die Zahl der weltweit angebotenen Seiten die Marke von 100 Millionen überschritten hat (Netcraft, 2006).

Die Vision des Semantic Web stammt von Tim Berners-Lee aus dem Jahr 1998 (Berners-Lee, 1998). Semantic Web soll die Kommunikation zwischen Programmen, sogenannten Software Agenten, verbessern. Dazu müssen die Informationen so aufbereitet werden, dass sie nicht nur von verschiedenen Programmen auf verschiedenen Computer fehlerfrei gelesen werden können, wie beispielsweise XML, sondern auch verstanden. Die Software Agenten sollen die Bedeutung von im Internet veröffentlichen Informationen begreifen und daraus automatisch Schlussfolgerungen ziehen können. Die „Überproduktion“ der Daten betrifft auch das Bauwesen. So werden während der Bauwerksüberwachung viele Daten benötigt oder auch produziert.

Diese Arbeit bildet eine Brücke zwischen der Informatik und dem Bauwesen. Dabei werden die Konzepte aus dem Semantic Web durch die Integration in einer modernen Entwicklungsumgebung an die Lösung der Probleme aus dem Bauwesen angepasst. Semantic Web kann zum Einen für die Suche der benötigten Webservices, sowie dazugehörigen Berechnungsmodellen zum Einsatz kommen. Zum Anderen können die Berechnungsergebnisse, sowie die benötigten Eingabedaten ebenfalls semantisch beschrieben werden und dadurch effektiv in weiteren Projekten genutzt werden.

Gliederung

Diese Arbeit ist in drei Kapitel aufgeteilt: im ersten Kapitel werden die theoretischen Grundlagen sowie die aktuellen Technologien und Standards kurz vorgestellt. Danach folgt das Kapitel der Synthese. Dieses beschreibt die während dieser Arbeit entdeckten Problemstellungen sowie erarbeiteten Methoden zu deren Lösung. Das letzte Kapitel beschreibt die praktische Umsetzung von in der Synthese erarbeiteten Methoden, dabei werden die aktuellen Technologien und Werkzeuge für die Anwendung in einem größeren Vorhaben diskutiert. Die Arbeit schließt mit einer Zusammenfassung und einem Ausblick.

1. Grundlagen

In diesem Kapitel werden die notwendigen Konzepte und Technologien kurz vorgestellt. Zuerst werden einige klassische Technologien für die Erstellung einer modernen verteilten Anwendung erläutert. Nachfolgend werden die darauf aufbauenden semantischen Konzepte und derzeit exisitierenden Werkzeuge und Standards vorgestellt.

1.1 Webservices

„Web Services sind aus der Software-Entwicklung praktisch nicht mehr wegzudenken – sie leisten in vielen Einsatzgebieten und unterschiedlichsten Branchen gute Dienste“

Thilo Frotscher (Frotscher, et al., 2007)

Webservices sind heute sehr verbreitet und werden in vielen Branchen eingesetzt. Der Erfolg von Webservices liegt in dem standardisierten Konzept, das durch die Web Services Interoperability Organisation (WS-I) durch die Veröffentlichung von Richtliniendokumenten festgelegt wird (Frotscher, et al., 2007). Die Standards für Webservices müssen einen Webservice aus verschiedenen Sichten beschreiben, damit die anderen Programme dieses finden, seine Schnittstelle begreifen und mit diesem kommunizieren können. Nachfolgend werden die drei derzeit am häufigsten dafür verwendeten Standards vorgestellt.

WSDL steht für ***Web Service Description Language*** und ist eine Sprache für die Beschreibung von Webservices. Die grundlegende Idee ist, dass ein WSDL-Dokument alle nötigen Informationen enthält die notwendig sind, um den Webservice aufzurufen. Zum einen beschreibt WSDL die abstrakte Schnittstelle eines Webservices zum anderen die Kommunikation mit dem Webservice. So kann beispielsweise ein WSDL-Dokument genutzt werden um den Source-Code (Service-Stub) für einen Webservice-Client automatisch zu erzeugen. Apache Axis liefert einige Codegeneratoren standardmäßig mit (Apa08). Die Nutzung der Generatoren wird oft durch die GUI erleichtert, zum Beispiel gibt es einige Eclipse Plugins für die Generierung eines Axis2 Webservices oder Clients. Der WSDL-Standard ermöglicht verschiedene Nachrichtenaustauschmechanismen. So könnte beispielsweise ein Service von sich aus eine Nachricht senden, die nicht durch eine unmittelbar zuvor enthaltende Nachricht eines Clients, sondern durch ein internes Ereignis ausgelöst wurde. Die Nachrichtenaustauschmechanismen werden mit Hilfe von Message Exchange Pattern (MEP) realisiert. In WSDL1.1 gibt es beispielsweise vier festdefinierte MEPs. (1) ***One-Way:*** Der Service empfängt eine Nachricht. (2) ***Request-Response:*** Der Service empfängt eine Nachricht und sendet daraufhin eine Nachricht zurück. (3) ***Solicit-Response:*** Der Service sendet eine Nachricht und erhält daraufhin eine Nachricht zurück. (4) ***Notification:*** Der Service sendet eine Nachricht und erwartet keine Antwort (Frotscher, et al., 2007). WSDL 2.0 unterstützt beliebige MEP’s, die MEP’s müssen lediglich mit einer URI eindeutig referenziert sein, wobei die acht gebräuchlichsten MEP’S in WSDL 2.0 vordefiniert sind. Außerdem unterstützt WSDL 2.0 ähnlich wie IDL[1] die Mehrfachvererbung, so kann ein Interface eines WSDL-Dokumentes von einem Interface eines anderen WSDL-Dokumentes abgeleitet werden. Im WSDL 2.0 Standard können im Vergleich zu WSDL 1.1 nicht nur XML-Schema Typen, sondern auch die funktionalen Schnittstellen wiederverwendet werden.

SOAP ist heute das wichtigste Kommunikationsprotokoll für Webservices. Ursprünglich stand der Name SOAP für ***Simple Object Access Protocol***, der Name kam aus der Welt von IIOP[2] und RMI[3], deswegen war auch das Wort Objekt so wichtig. Ab Version 1.2 ist SOAP ein einfacher

[1] IDL - Interface Definition Language

[2] IIOP - Internet Inter-ORB Protocol

[3] RMI - Remote Method Invocation

Name und kein Akronym mehr, dieser wird beibehalten, weil er die Technologie populär gemacht hat (Frotscher, et al., 2007). SOAP ist ein XML-basiertes Protokoll und besitzt deswegen eine große Interoperabilität. So kann beispielsweise ein C++ Programm unter Unix einen in Java entwickelten und auf dem Apache Tomcat unter Windows laufenden Webservice aufrufen. SOAP wird oft in Verbindung mit dem HTTP-Protokoll zusammen genutzt, das von meisten Firewalls durchgelassen wird. Jedoch ist die Nutzung mit anderen Transportprotokollen wie TCP[4] oder SMTP[5] möglich. SOAP besitzt einen eingebauten Fault Mechanismus, dadurch kann auf der Client-Seite genau festgestellt werden wo der Fehler entstanden ist. Ab SOAP 1.2 sind fehlerspezifische Informationen in der SOAP Nachricht selbst zu finden und nicht mehr wie bei SOAP 1.1 in HTTP-Header, so hat SOAP 1.1 zum Beispiel die Fehler durch HTTP-Status-Code 500 (Internal Server Error) beantwortet (SOA08). SOAP 1.2 beantwortet die Fehler mit einer SOAP-Nachricht in einem erfolgreichen HTTP-Response (HTTP-Status-Code 200), die die Fehlerbeschreibung in XML-Format enthält. Dadurch ist die Trennung zwischen der Transport- und Anwendungslogik viel deutlicher geworden.

Universal Description, Discovery and Integration (UDDI) (Bellwood, et al., 2003) bietet eine Schnittstelle, die es Businesspartnern und anderen Serviceanbietern ermöglicht, dynamisch die in der UDDI registrierten Webservices zu suchen und zu veröffentlichen. UDDI wurde ursprünglich von großen Firmen wie Microsoft, IBM, Ariba entwickelt. Die UDDI Spezifikation bietet eine API für die Suche und Veröffentlichung von Webservices. Die UDDI Business Registry ist eine vollständige Implementierung dieser Spezifikation und wurde ab Mai 2001 von Microsoft und IBM in Betrieb genommen (Cerami, 2002). Eine UDDI-Implementierung ist ein Webservice, deren Schnittstelle die UDDI-API Spezifikation implementiert (Fensel, et al., 2007). Die veröffentlichten Informationen in einer UDDI werden in drei Kategorien aufgeteilt:

White pages

Allgemeine Informationen über eine Firma, zum Beispiel Firmenname (business name), Beschreibung der Gewerbetätigkeit (business description), Postadresse.

Yellow pages

Enthält allgemeine Klassifikation entweder für das Angebot einer Firma oder eines Services. Zum Beispiel können hier standardisierte Codes aus der Industrie, ein Produkt oder geografische Standard Taxonomien enthalten sein.

Green pages

Enthält technische Informationen über einen Webservice, also eine Adresse für dessen Aufruf, sowie einen externen Link zur technischen Dokumentation.

(Cerami, 2002)

Die veröffentlichte Information über einen Webservice wird von dem Webservice-Anbieter erzeugt und erleichtert die Suche nach dem gewünschten Webservice.

Die UDDI hat vier Arten von Informations-Datenstrukturen (Web081): (1) ***businessEntity*** beschreibt den Webserviceanbieter, (2) ***businessService*** charakterisiert Webservices hinsichtlich des Service-Angebots, (3) ***bindingTemplate*** enthält detaillierte technische

[4] TCP – Transmission Control Protocol

[5] Simple Mail Transfer Protocol

Informationen zur Nutzung des Webservices, (4) ***tModel*** ist ein generischer Container, der die detaillierten Service-Informationen enthält. Der Zusammenhang zwischen einzelnen UDDI-Datenstrukturen ist in der Abbildung 40 (siehe Anhang) veranschaulicht. UDDI war ein Misserfolg im Bereich von öffentlichen Webservices und URB wurde von großen Firmen eingestellt, nur die kleinen Firmen bieten die UDDI-Schnittstelle als Zugang für ihre Service Repository. Heute wird UDDI hauptsächlich im Intranet[6] und Extranet[7] verwendet (Fensel, et al., 2007). Das Hauptproblem einer UDDI ist ihre statische Struktur. Eine solche Struktur macht es unmöglich, eine schnelle Anpassung einer Anwendung an die neuen Anforderungen vorzunehmen. Außerdem basiert UDDI auf einer Textsuche und nutzt somit nicht die Vorteile der semantischen Beschreibung von Informationen. So existiert beispielsweise keine indirekte Suche, wo die Webservices basierend auf logischen Schlussfolgerungen aus bereits in der UDDI enthaltendem Wissen schlussgefolgert werden können (Cardoso, 2007).

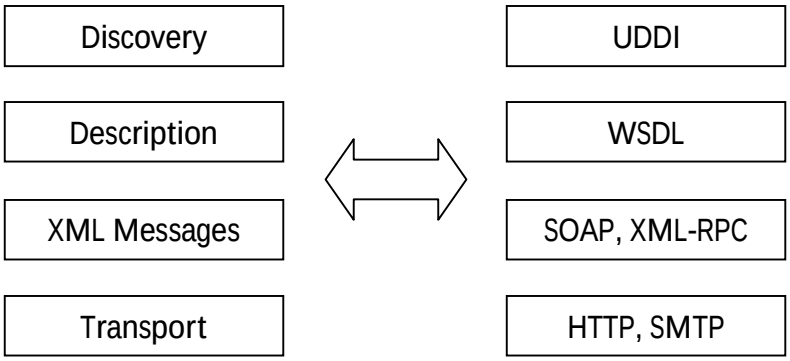

Abbildung 1 Webservice Protokoll Stak

Derzeit gibt es zwei Methoden einen Webservice zu entwickeln. Jede dieser Methoden hat ihre Vorteile und Nachteile, einige davon sind in dem unten stehenden Zitat deutlich aufgeführt.

„Wenn man bei der Entwicklung mit XML Schema-Datentypen startet, so ist es leicht, diese auf plattformspezifische Datentypen abzubilden, da entsprechende Abbildungsregeln bereits existieren. Startet man dagegen mit plattform- oder sprachspezifischen Datentypen, so kann es passieren, dass keine standardisierte Abbildung nach XML besteht und somit andere Kommunikationspartner Schwierigkeiten haben werden, mit diesen umzugehen.“

(Frotscher, et al., 2007)(Seite 57)

Beim ***Code-First-Ansatz***[8] wird zuerst der Programcode implementiert und erst dann mit Hilfe eines Generierungswerkzeugs ein WSDL-Dokument dafür erstellt (Abbildung 2). So generiert beispielsweise Axis2-JavaToWSDL-Generator automatisch eine WSDL Datei für eine Java-Klasse. So ein Generator spart viel Entwicklungszeit, da er nicht nur die Operationsbeschreibung generiert, sondern auch die komplexen Java-Klassen in XML-Schema Typen abbildet. Der

[6] Intranet ist ein unternehmensinternes Rechnernetzwerk

[7] Extranet ist eine Erweiterung von Intranets, es ermöglicht die Verwendung von Ressourcen von einer Gruppe externer Benutzer

[8] Code-First-Ansatz wird auch als Top-Down-Methode bezeichnet

JavaToWSDL-Generator funktioniert leider nicht immer wie erwartet, und manchmal muss die WSDL Beschreibung per Hand angepasst oder sogar erstellt werden (Frotscher, et al., 2007) (Seite 55). Das schöne am Code-First-Ansatz ist, dass man sich um die komplexen Standards eines Webservices gar nicht kümmern muss, wie zum Beispiel die Generierung eines WSDL-Dokumentes oder Abbildung eines Java Typs in einen XML-Schema Typ. Dieser Ansatz bringt auch Vorteile, wenn die Schnittstelle von der Java-Klasse, die die Anwendungslogik für den Webservice enthält, verändert wurde. Dann kann ganz schnell ein neues WSDL-Dokument generiert und veröffentlicht werden. Das Problem dieses Ansatzes liegt in der schlechten Interoperabilität mit fremden Technologien, da jeder Webservice Generator seine eigene Methode zur Abbildung der Klassen einer Programmiersprache in die XSD-Datentypen verfolgt. Die Schärfe dieses Problems nahm etwas ab, nachdem das WS-I (Web Services Interoperability Forum) den Basic Profile, ein Richtliniendokument für die Verwendung von Webservice-Technologien, veröffentlicht hat. In der Praxis existieren bereits XML-Schemata die möglichst große Interoperabilität zwischen verschiedenen Vertragspartnern gewährleisten. Die Geschäftspartner können ihre Software gegen diesen gemeinsamen Vertrag validieren.

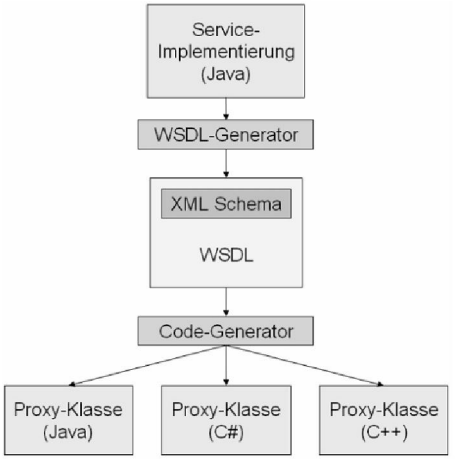

Abbildung 2 Ablaufschema beim Code-First-Ansatz (Frotscher, et al., 2007)

„Leider ist jedoch festzustellen, dass keines der vorhandenen Werkzeuge zur Erstellung von WSDL-Dokumenten frei von Fehlern ist. Dies liegt sicherlich wiederum an der Komplexität der Spezifikation und den vielen darin versteckten Stolperfallen."

(Frotscher, et al., 2007) (Seite 60)

Eine andere Methode zum Entwicklen eines Webservices ist der ***Contract-First-Ansatz***[9]***.*** Beim Contract-First-Ansatz wird zuerst die Spezifikation eines Webservices in Form eines WSDL-Dokumentes erstellt, dabei werden in das WSDL-Dokument die gemeinsam genutzten XML-Schema Typen importiert und eventuell neue definiert (Abbildung 3). Die die Operationen von Schnittstellen werden mit Hilfe dieser Typen definiert (Input, Output). Ein auf solche Weise erstelltes WSDL-Dokument sollte mit dem Analyse-Tool des WS-I auf seine Konformität mit dem Basic Profile überprüft werden, bevor aus dem WSDL-Dokument der Programmcode(Skeleton)

[9] Contract-First-Ansatz wird auch als Buttom-Up-Methode bezeichnet

generiert wird. Wenn die Analyse erfolgreich ist, kann mit der Implementierung des erstellten WSDL-Dokumentes fortgefahren und der Skeleton generiert werden. Die Contract-First Methode erfordert ein tieferes Verständnis von XML-Schema und WSDL-Standard sowie ein Verständnis von WS-I Basic Profiles. Leider ist die Toolunterstützung für Contract-First noch recht schlecht. Ein Entwickler steht damit der Komplexität von XML-Schema und WSDL Spezifikationen ohne ausreichende Werkzeugunterstützung gegenüber (Frotscher, et al., 2007).

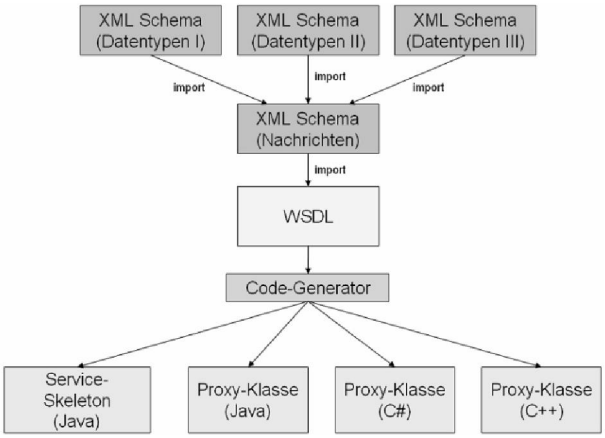

Abbildung 3 Auflaufschema beim Contract-First-Ansatz (Frotscher, et al., 2007)

1.2 Webservice - Orchestrierung

„Seien es internationale Hotelreservierungssysteme mit einer Vielzahl angebundener Hotelketten und Vertriebspartner, Systeme im Finanzdienstleistungsbereich, in der Logistikbranche oder verschiedenste Stellen im öffentlichen Sektor (Behörden, Ministerien usw.) – die Einsatzgebiete sind ausgesprochen vielfältig“

(Frotscher, et al., 2007)

WS-BPEL (Web service Business Process Execution Language) (Alves, et al., 2007) existierte anfangs unter dem Namen BPEL4WS, erst später wurde es in WS-BPEL umbenannt. Diese Spezifikation ist allgemein besser als BPEL bekannt (By Sanjiva Weerawarana, 2005). BPEL ist eine Architektursprache für Webservices. Sie ermöglicht eine Orchestrierung von Webservices und somit die Erstellung von Business-Prozessen. BPEL basiert auf mehreren Standarten wie SOAP, WSDL und XML(Vasiliev, 2007). Ein BPEL Prozess enthält oft mehrere Webservices, die nacheinander oder parallel aufgerufen werden. BPEL kommuniziert mit Webservices ausschließlich über SOAP-Nachrichten. So wird ein Webservice mit Hilfe eines in dem BPEL-Prozess generierten SOAP-Request aufgerufen. Die Antwort von dem Webservice ist ein SOAP-Response. Dieser wird in der Geschäftsprozesslogik eines BPEL-Prozesses weiterverarbeitet. Ein BPEL-Prozess kann aus einem anderen BPEL-Prozess aufgerufen (Komposition, Dekomposition) oder selbst als ein Webservice veröffentlicht werden. Die Daten in einem BPEL Prozess werden in sogenannten Assign[10] Activities manipuliert. Die meisten BPEL

[10] Assign (dt. zuordnen)

Variablen entstehen aus WSDL Message Typen oder importierten XML-Schema Typen. Die Verarbeitung von SOAP-Nachrichten geschieht standardmäßig mit Hilfe von XPath (Weerawarana, et al., 2005), jedoch können auch andere Technologien eingesetzt werden, wie beispielsweise XSLT-Transformation. So können beispielsweise die Daten aus einem SOAP-Request an einen BPEL-Prozess für die Generierung eines SOAP-Requests eines orchestrierten Webservices verwendet werden. Die ActiveBPEL Engine nutzt die XSLT Transformation, wodurch viele Probleme auf eine elegantere und effizientere Art und Weise gelöst werden können. So kann beispielsweise durch XSLT-Transformation die Performance gewaltig gesteigert werden. Ebenso kann die Transformationslogik geringere Komplexität als bei Verwendung von XPath haben. BPEL unterstützt die Verwendung von abstrakten und ausführbaren Prozessen, so kann eine Firma einen abstrakten Prozess, der einen ausführbaren Prozess beschreibt an seine Business Partner ausliefern. Dadurch bleibt die interne Logik eines ausführbaren Prozesses vor den Business Partnern versteckt und trotzdem können diese die Funktionsweise eines ausführbaren Prozesses nachvollziehen. Die Deklaration von zu verwendenden Webservices oder BPEL-Unterprozessen erfolgt über sogenannte Partnerlinks. Die Partnerlink Typen repräsentieren die Connectoren zu existierenden Webservices oder anderen BPEL Prozessen, über die die gesamte Kommunikation eines BPEL-Prozesses mit der Außenwelt erfolgt. Die Bindings und Ports eines Partnerlinks können zur Deployment-Zeit festgelegt werden, was einen BPEL-Prozess mehr anpassbar und portabel macht (Weerawarana, et al., 2005). Falls ein in dem BPEL-Prozess verwendeter Webservice auf einem anderen Server deployed wird, kann der BPEL-Prozess durch das Verändern des Deployment Descriptors wieder funktionstüchtig gemacht werden, da in einem statischen Deployment Descriptor die Bindings und Ports eines Partnerlink festgelegt sind. Es gibt vier Bindungsarten:

Static design time binding

Die Bindung ist ein Teil der Geschäftslogik. Diese Bindungsart ist nicht besonders flexibel.

Static deployment time binding

Die Endpoints werden zu Deployment Zeit festgelegt.

Dynamic binding using lookups

Ein Partnerlink enthält die Anforderungen, die ein Endpoint haben soll. Danach werden zur Laufzeit oder Deploymentzeit die Webservices gesucht, die diese Anforderungen erfüllen. Die Anforderungen sind zum Beispiel: Quality-of-service, transactional capabilities, enthaltene Funktionalität.

Dynamic binding using passed-in endpoints

Der Endpoint wird dem auszuführenden Prozess als eine Variable übergeben. Der Prozess nutzt diese Variable um den Endpoint zu setzen und erst dann ruft er über dieses den Webservice auf. Diese Bindungsart wird auch als Lazy Bindung bezeichnet.

Open Service Process Platform 2.0 (OSPP)(Habich, et al., 2008) ist eine neuartige Workflow Entwicklungsumgebung, die auch eine eigene Webservice Orchestrierungssprache besitzt. Im Gegensatz zu WS-BPEL hat OSPP viele technologische Vorteile. So beispielsweise bietet die OSPP das Konzept der Data-Gray-Box für Webservices, wodurch der Datentransfer zwischen Webservices viel effizienter gehandhabt wird. Ebenso ermöglicht OSPP die Prozesse zur Laufzeit zu vervollständigen. Das ist manchmal notwendig, wenn beispielsweise eine Exception des unbekannten Datentyps zur Laufzeit geworfen wird. In diesem Fall muss der Prozess um neue Logik erweitert werden damit diese Exception vermieden oder behandelt werden kann (Habich,

et al., 2008). OSPP ermöglicht im Gegensatz zu einer gewöhnlichen BPEL Entwicklungsumgebung nicht nur die Erstellung der Workflow Prozesse sondern auch ihre Ausführung. Dazu kommt, dass die OSPP eine Webanwendung ist und somit von überall und jederzeit verfügbar ist. So können mehrere Nutzer die Prozesse gemeinsam erstellen oder nutzen, was wiederum eine engere Zusammenarbeit ermöglicht. OSPP kann relativ einfach mit neuen Plugins erweitert und somit an die speziellen Bedürfnisse einer Institution angepasst werden. Außerdem bietet OSPP die Repositorien für Webservices und Prozesse, sowie die Rechteverwaltung und kann als eine universelle Entwicklungsumgebung für die Erstellung von Prozessen in einer Firma genutzt werden.

1.3 Semantic Web

„ The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation."

(Berners-Lee, 1998)

Das ***World Wide Web*** (auch kurz ***Web*** genannt) ist heute nicht mehr wegzudenken. Das Web enthält eine unvorstellbare Menge an Information, die ständig weiter wächst. Wegen der dezentralen Web Struktur ist diese Information überall in der realen Welt verteilt. Das Web ermöglicht den Zugriff auf diese Informationen aus jedem Ort und zu jeder Zeit. Die meisten Informationen liegen in nur für Menschen lesbarem Format vor und können von einer Maschine nicht automatisch analysiert werden. Das Problem von Web liegt darin, dass es mehr Informationen enthält als die Menschen verarbeiten können. Die Informationen sind jedoch erst dann nützlich, wenn diese in einer Entscheidung berücksichtig werden können. Der Einsatz von Suchmaschinen wie Google oder Yahoo leistet in der Verarbeitung solcher Informationen erstaunliche Ergebnisse. Jedoch basieren die statischen Techniken solcher Suchmaschinen auf dem Durchsuchen des Textes, dabei wird die Bedeutung des Textes nicht berücksichtigt. So beispielsweise liefert die Suche nach dem Schlüsselwort „Kohl" die Ergebnisse nach dem Gemüse aber auch nach dem Altbundeskanzler (Hitzler, et al., 2007). Das Semantic Web wird das bestehende Web erweitern und dem Benutzer neue Suchdienste in vielen Bereichen anbieten. Die dafür nötigen Standards wie RDF(S), OWL wurden vom W3C Konsortium bereits definiert und werden ständig weiter entwickelt. Semantic Web soll die Informationen so repräsentieren, dass die Maschinen mit diesen Informationen auf eine Art und Weise umgehen können, die aus menschlicher Sicht nützlich und sinnvoll erscheint (Berners-Lee, 1998). Es ist nicht das Ziel von Semantic Web die Informationen mit Hilfe von künstlicher Intelligenz zu verarbeiten, obwohl die neuen Informationen durchaus aus existierenden mit Hilfe einer Inferenzmaschine abgeleitet werden können. Trotz starker Motivation und Versprechen die das Semantic Web macht ist es noch eine Vision und existiert nicht wirklich, weil die dafür nötigen Werkzeuge fehlen:

"Generally considered a subset of the next-generation Web technologies referred to as Web 3.0, the semantic Web doesn't really exist yet, largely because the available tools aren't up to the task."(Claburn, 2007)

Viele Technologien, die das Semantic Web ermöglichen existieren bereits und sind reif für die Praxis, jedoch sind diese kompliziert und erfordern Fachwissen. Die meisten Informationen werden jedoch von Computernutzer erzeugt, die keine Ahnung von Prädikatenlogik und von der

Softwareentwicklung haben. Diese Informationen werden oft nicht ausreichend semantisch beschrieben und können von Maschinen nicht interpretiert werden. Damit Semantic Web möglich ist, müssen die Informationen semantisch beschrieben werden.

„Um durch automatisiertes Schlussfolgern möglichst viel Information zu gewinnen, sind ausdrucksstarke Beschreibungslogiken nötig. Ist die Ausdrucksmächtigkeit jedoch zu groß gewählt, so ist die Logik nicht mehr entscheidbar. Ohne terminierende und e ziente Inferenz-Algorithmen ist eine Logik nicht für das Semantic Web geeignet. Daher müssen Logikstufen verwendet werden, die einen guten Kompromiss zwischen Ausdrucksmächtigkeit und Entscheidbarkeit darstellen“

Carsten Angeli (Bauer, 2008)

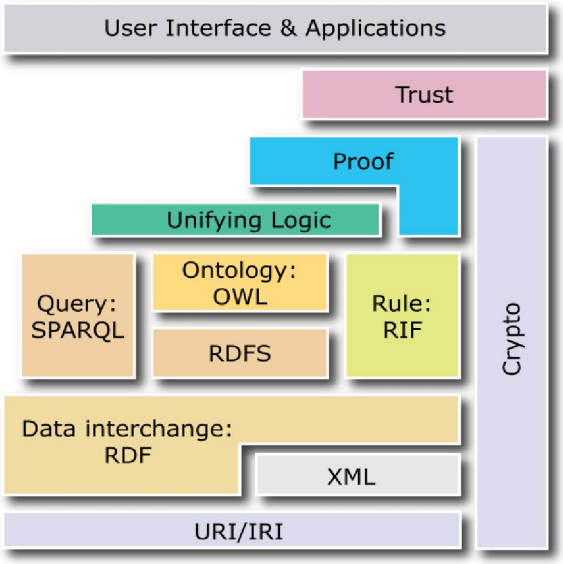

Abbildung 4 Schichtenmodel (Quelle: http://www.w3.org/2007/03/layerCake.png)

Die semantische Beschreibung wird durch das Zusammenspiel von vielen Standards ermöglicht. In der Abbildung 4 ist das vom W3C definierte Technologie Schichtenmodel abgebildet. Insgesamt gibt es sieben Schichten, die aufeinander aufbauen. Nachfolgend werden die einzelnen Schichten aus der Abbildung 4 kurz vorgestellt: ***URIs***[11] sind wichtig für die Kommunikation zwischen Maschinen, so können beispielsweise die Ressourcen eindeutig identifiziert werden. Zwei syntaktisch gleiche Begriffe, mit unterschiedlicher Semantik, können durch verschiedene URIs eindeutig referenziert werden. ***XML*** ermöglicht eine einheitliche Darstellung von verschiedenen Daten und verbessert somit die Interoperabilität zwischen verschiedenen Tools. Außerdem kann eine XML-Dokumentenstruktur für beliebige Anwendung leicht angepasst und erweitert werden. Das ***Resource Description Framework (RDF)*** ist eine Formale Sprache für die Beschreibung von Informationen. Mit Hilfe von RDF können Informationen im Web ausgetauscht werden, ohne dass ihre Bedeutung dabei verloren geht. Heute wird RDF in vielen Anwendungen verwendet, zum Beispiel das Format RSS 1.0 (RDF Site Summary) basiert auf RDF und ermöglicht ein Abonnement von Nachrichten (Hitzler, et al., 2007). RDF enthält Informationen in Form eines RDF-Graphen. Der Graph kann mit Hilfe von

[11] URI – Unified Ressource Identifier

sogenannten ***Tripels*** oder ***Turtles*** beschrieben werden. Damit diese Daten auf jedem System eingelesen werden können, kann RDF auch mit Hilfe von XML ausgedrückt werden. Dabei beschreibt XML nur die Struktur von RDF, die Semantik ist in RDF-Termen enthalten. XML kann von einem beliebigen XML-Parser eingelesen werden und bietet somit eine gute Interoperabilität. In RDF gibt es nur einen einfachen Datentyp Literal, dieser repräsentiert eine Zeichenkette. In Anwendungen werden jedoch oft allgemein bekannte Datentypen benötigt. Deswegen wurde ***RDFS*** definiert, das die Verwendung von XML-Schema Typen ermöglicht, sowie die Definition und Vererbung eigener RDF-Datentypen(Klassen). RDF(S) ist zur Modellierung einfacher Ontologien geeignet und ist für die Darstellung komplexer Zusammenhänge nicht ausreichend. So lassen sich beispielsweise natürlichsprachliche Sätze nicht hinreichend genau in RDF(S) modellieren (Hitzler, et al., 2007). Die ***Web Ontology Language (OWL)*** wurde vom W3C als Ontologiesprache veröffentlicht. In der Entwicklung von OWL wurde ein Kompromiss zwischen dem effizienten Schlussfolgern und der Ausdrucksstärke einer Sprache erlangt. Normalerweise besitzen die Sprachen mit hoher Ausdruckstärke hohe Schlussfolgerungskomplexität. Es gibt drei verschiedene Teilsprachen OWL Lite, OWL DL, OWL Full (siehe Abbildung 5):

OWL Full enthält OWL DL, OWL Lite und RDFS. OWL Full ist sehr ausdrucksstark, deswegen erfordert das Schlussfolgern oft sehr viel Zeit. Manche Aspekte der Semantik sind jedoch aus logischem Blickwinkel problematisch (Hitzler, et al., 2007). OWL Full ist nicht entscheidbar und wird durch aktuelle Softwarewerkzeuge nur begrenzt unterstützt.

OWL DL (Description Logic) ist eine Teilsprache von OWL Full, sie stellt einen Kompromiss zwischen komplexen Ausdruckskonstrukten und dem effizienten Schlussfolgern dar. OWL DL ist entscheidbar und wird von den aktuellen Werkzeugen fast vollständig unterstützt. So unterstützt beispielsweise die Inferenzmaschine Pellet(pel08) vollständig die OWL DL. OWL DL hat eine Komplexität NExpTime(worst-case) was das Schlussfolgern reif für die Praxis macht.

OWL Lite ist eine Teilsprache von OWL DL und OWL Full. OWL Lite ist entscheidbar und hat eine Komplexität ExpTime(worst-case) (Hitzler, et al., 2007). OWL Lite ist besonders gut für die Erschaffung von einfachen Taxonomien geeignet.

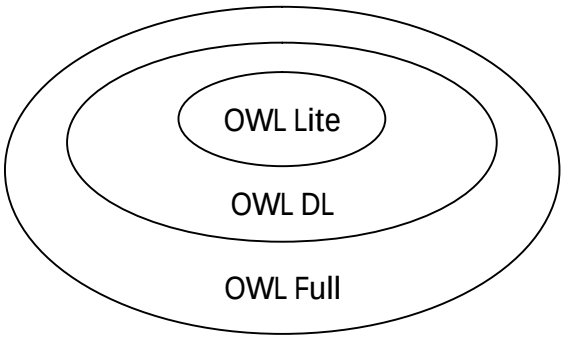

Abbildung 5 OWL Teilsprachen

SPARQL Protocol und RDF Query Language (SPARQL) (Prud'hommeaux, et al., 2008) ist nun nach vielen Jahren endlich am 15. Januar 2008 zur endgültigen Recommendation avanciert. SPARQL hat sich in den letzten Jahren als eine geeignete Anfragesprache für RDF herauskristallisiert. Sie ermöglicht es, Anfragen durch benutzerdefinierte Kriterien zu filtern, und dadurch die Ergebnismenge einzuschränken. Eine SPARQL-Anfrage basiert auf der Idee des Pattern Matchings, so wird in einer Anfrage ein Muster beschrieben, mit dem ein RDF-Teilgraph

übereinstimmen muss. Die Ergebnismenge wird aus gefundenen Teilgraphen generiert. Die SPARQL kann in die relationale Algebra[12] übersetzt werden, wodurch viele Vorteile entstehen: (1) Die Anfragen lassen sich besser planen und optimieren, (2) Die in der relationalen Algebra entwickelten und erprobten Algorithmen können auch in der SPARQL eingesetzt werden (Hitzler, et al., 2007).

Eine ***Inferenzmaschine (engl. reasoner)*** ist ein Programm, das Inferenzregeln zum Ableiten neuen Wissens verwendet. Diese Regeln sind Bestandteil eines Kalküls, welches die Semantik für das Schlussfolgern genau beschreibt. Mit Hilfe dieser Semantik kann eine Inferenzmaschine die logischen Formeln lösen. Ob diese Formeln von einer Inferenzmaschine gelöst werden können, hängt von zwei Aspekten ab: zum Einen ob die Formel gültig ist und zum anderen ob diese erfüllt werden kann. Das ***Schlussfolgern (engl. reasoning)*** ist wichtiger Bestandteil des Semantic Web. Es gibt viele Algorithmen für das Schlussfolgern aus existierenden Fakten. Der wohl bekannteste ist der Rete-Algorithmus, der in abgewandelter Form die Grundlage für viele bekannte und sich im Einsatz befindende Expertensysteme bildet (Bauer, 2008). Eine Inferenzmaschine sollte folgende Anforderungen erfüllen:

1. Korrektheit: Eine Inferenzmaschine sollte nur gültige Aussagen ableiten
2. Vollständigkeit: Alle möglichen gültigen Aussagen ableiten
3. Entscheidbarkeit: Entscheidung über die Gültigkeit einer Aussage in endlicher Zeit
4. Effizienz: Eine effiziente Bearbeitung einer Anfrage
5. Skalierbarkeit: Inferenzmaschinen sollten mit stetig größer werdender Wissensbasis umgehen können
6. Adaptive Performance: Semantic Web ist sehr dynamisch, das neue Wissen kommt hinzu, das alte verschwindet
7. Robustheit: Inkonsistenzen wie widersprüchliche Aussagen oder nicht funktionierende Links kommen im Web oft vor. Solche Ausnahmen sollten die Funktionsfähigkeit einer Inferenzmaschine nicht beeinflussen.

(Endres, et al., 2006)

1.4 Semantic Webservices

Klassische Webservice Standards wie WSDL ermöglichen nur eine syntaktische Beschreibung eines Services. So ist es möglich zu beschreiben wie ein Service aufgerufen werden kann. Leider ist es nicht möglich mit herkömmlichen Standards zu beschreiben, was der Service macht oder in welcher Reihenfolge die Operationen aufgerufen werden sollen. Solche Beschreibungen können zwar in Textform in der UDDI gespeichert werden, jedoch sind diese nicht maschinenlesbar (Fensel, et al., 2007). Semantisch beschriebenen Webservices können nicht nur schneller gefunden werden sondern bieten viele neue Möglichkeiten an. Abbildung 7 veranschaulicht den ***Lifecycle eines Semantic Webservices***. Er enthält fünf Schritte:

1. Advertisement:

Advertisement ist der Prozess der Veröffentlichung der semantischen Beschreibung eines Webservices in der Service Registry. Es gibt zwei Arten der Veröffentlichung eines Webservices:

[12] Relationale Algebra ist eine formale abstrakte Sprache um Anfragen besser analysieren zu können

zum einen kann die semantische Beschreibung eines Webservices im Web veröffentlicht werden und eine Suchmaschine findet diese und trägt sie in ihre Registry ein, zum anderen registriert der Service-Anbieter selbst die veröffentlichte semantische Beschreibung in der gewünschten Service Registry.

2. Discovery:

Hier werden Webservices gesucht die den Forderungen der Suchanfrage entsprechen. Wobei hier nicht wie in einer klassischen Registry nur atomare Bausteine gesucht werden, sondern auch Prozessketten. Jeder Prozess ***(Composite Service)*** kann mehrere Semantic Services orchestrieren. Zum Beispiel Prozess-1 (Abbildung 6) orchestriert folgende ***Semantic Services***: A, B, C. Die orchestrierten Services können wiederum Composite oder Atomic Services sein. Ein ***Atomic Service*** orchestriert im Gegensatz zum ***Composite Service*** keine weiteren Services, sonder beschreibt normalerweise eine implementierte Operation eines klassischen Webservices. Da die Workflow Prozesse (Composite Service) semantisch beschrieben sind, können diese zur Suche noch nicht explizit existierender Funktionalität herangezogen werden. So können Teile von mehreren Prozessen zu einem neuen Prozess zusammengefügt werden. Zum Beispiel sind in Abbildung 6 drei Prozesse abgebildet. Prozess-1 und Prozess-2 sind Composite Services**.** Nun stellt ein Nutzer eine Anfrage an die Registry, dass er einen Webservice braucht, der beispielsweise die Input-Variablen eines ***Atomic-Service*** A und die Output-Variablen des Atomic-Service D braucht. Der Such-Algorithmus durchsucht die registrierten Services und findet mehrere Möglichkeiten so einen Prozess aus bereits existierenden Prozessen durch Komposition zu erstellen. Eine Möglichkeit wäre beispielsweise durch die Ausnutzung der Verknüpfung von A nach B in dem Prozess-1 und von B nach D aus dem Prozess-2 einen neuen Prozess (Prozess-3) zu erstellen, bei dem die Prozessketten aus Prozess-1 und -2 verknüpft sind.

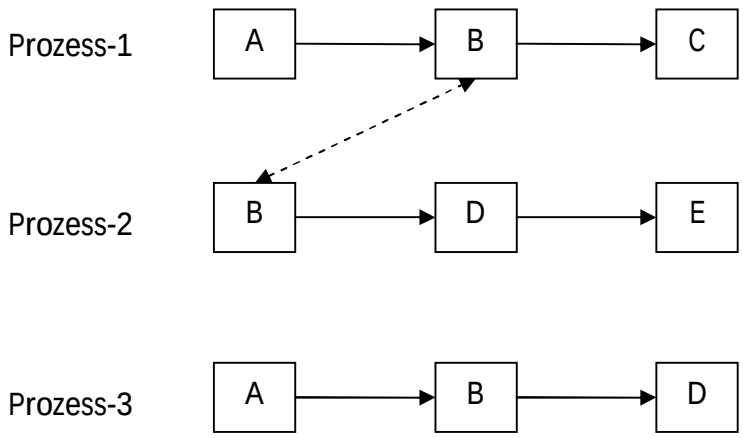

Abbildung 6 Semantische Prozess-Suche

3. Selection:

Selection ist der Prozess der Auswahl von Webservices aus zuvor in Discovery gefundenen Webservices nach programmspezifischen Kriterien. Zum Beispiel kann der Suchalgorithmus mehrere mögliche Services finden, die den gestellten Anforderungen entsprechen. Nun wird in der Selection ausgewählt, welcher Service am besten für die Beantwortung einer Anfrage

geeignet ist. Zum Beispiel, wenn der Suchalgorithmus mehrere mögliche Prozesse findet, die aus Prozessketten anderer Prozesse bestehen, könnte der Dijkstra Algorithmus verwendet werden, um den Prozess mit dem kürzesten Pfad aus allen möglichen zusammengestellten Prozessen auszuwählen.

4. Composition:

In diesem Schritt wird aus der semantischen Beschreibung eines Webservices ein ausführbarer Prozess erstellt. Zum Beispiel OWL-S Editor (The082) (ist ein Plugin für Protege) kann aktuell die Atomic Prozesse ausführen. Die Entwickler versprechen jedoch auch die Ausführung von Composite Prozessen zu implementieren: *"but we aim to support composite as well as atomic processes, and users will be able to take the results returned from services and add them into the Protégé knowledge base."* (Elenius, et al.).

5. Invocation:

Der während der Composition erstellte Prozess wird ausführt.

(Cardoso, 2007)

Nach der Invocation kann nun wieder zum Advertisment übergegangen werden, denn der während der Selection gefundene Prozess könnte im Semantic Web als neue Information veröffentlicht und für weitere Suchen verwendet werden.

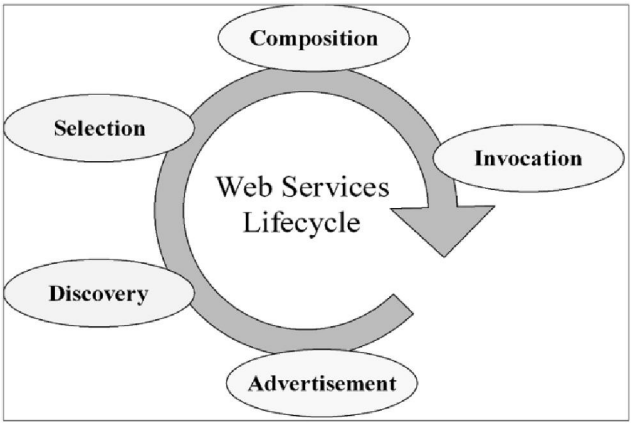

Abbildung 7 Semantic Webservice Lifecycle(Cardoso, 2007)

In (Fensel, et al., 2007) wird unter einem Semantic Webservice eine Kombination der semantischen Beschreibung von Daten und der semantischen Beschreibung von Logik verstanden. So können beispielsweise die semantisch beschriebenen Daten als Input für einen semantisch beschriebenen Webservice dienen. In dieser Arbeit wird dieses Konzept verfolgt und mit Konzepten aus der Bauinformatik zu einem akademischen Prototyp ausgebaut. In der Zukunft ist zu erwarten, dass die Semantic Webservices das Web in eine globale Infrastruktur für verteilte Systeme umwandeln und eine Grundlage für automatische Erstellung von Business Prozessen bilden werden. Semantic Web wird nicht nur ein Ort sein, wo die semantisch beschriebenen Informationen leicht gefunden werden können, sondern ein Ort wo alles global berechnet werden kann (Fensel, et al., 2007).

1.5 Semantic Webservice Registry

"The real problem with exact matching methods is that they cannot fully capture the intentional semantics of services or service requests."

(Cardoso, 2007) (Seite 246)

Die Verbreitung von Webservices entwickelt eine eigene Dynamik. Genauso wie die Serviceanbieter auf den Markt kommen und gehen, entstehen und verschwinden die Webservices im WWW. Das hat zur Entwicklung von Technologien geführt, die auf solche Ereignisse reagieren können und sich an die neue Situation dynamisch anpassen können. Eine solche Methode stammt aus Semantic Web. Sie verwendet semantische Beschreibung von Webservices mittels Ontologien und einem darauf aufbauenden System zur Interpretation und Weiterverarbeitung von diesen (Bauer, 2008). Semantic Web bietet neue Möglichkeiten Webservices zu suchen. Viele Technologien wie RDF/OWL wurden in letzter Zeit stark weiterentwickelt und bieten zum Teil eine ausgereifte Basis für den Einsatz in der Praxis. Eine ***semantische Service Registry*** (wird im Weiteren als ***Service Registry*** bezeichnet) ist eine Implementierung des Verzeichnisdienstes aus der UDDI und entspricht den Gelben Seiten (Abbildung 8). Im Gegensatz zu Gelben Seiten aus der UDDI, die die Beschreibung eines Webservices intern abspeichert, verweist die semantische Service Registry auf die im WWW veröffentlichte semantische Beschreibung eines Webservices. Eine der Schwächen von UDDI ist die Verwendung von XML als statisches Datenmodell. XML garantiert zwar syntaktische Interoperabilität, ermöglich aber nicht die semantische Beschreibung eines Webservices. Außerdem können die Maschinen XML nicht verstehen (Srinivasan, et al., 2005). Laut (Cardoso, 2007) ist das größte Problem von UDDI, dass die Matching Algorithmen nicht vollständig die Semantik von beschriebenen Webservices verstehen können. Die Suche in einer klassischen UDDI ist syntaxbasiert und führt bei einer großen Anzahl der registrierten Webservices selten zu sinnvollen Ergebnissen. Eine semantische Suche von Webservices stützt sich auf die Verwendung von Ontologien. So werden beispielsweise ***Service Advertisments*** mit Hilfe von Ontologien erstellt. Das Service Advertisment enthält die semantische Beschreibung des Webservices, also ***was*** der Service ***macht***, ***wie*** der Webservice ***aufgerufen*** werden kann, und ***wie*** er ***arbeitet*** (Cardoso, 2007). Die Semantische Service Registry nutzt für die Suche einen sogenannten ***Matchmaking Algorithmus*** (Suchalgorithmus, siehe Kapitel 1.8), der normalerweise mit der Service Registry fest gekoppelt ist. Der Matchmaking Algorithmus ist für die Verarbeitung von ***Suchanfragen (Service Request)*** verantwortlich. Diese semantischen Suchalgorithmen sind normalerweise viel intelligenter als ihre syntaxbasierten Vorgänger aus der UDDI. Die Suchanfragen werden von einem ***Service Requester*** gestellt, der ein beliebiges Programm sein kann. Ein Request kann in einer beliebigen Sprache formuliert werden, er soll lediglich die Anforderungen einer Anfrage enthalten. Im Idealfall ist ein Request in einer semantischen Beschreibungssprache wie OWL-DL formuliert. In der Realität wird jedoch ein Request in einer syntaxbasierten Sprache wie XML formuliert und es wird eine Abbildungsschicht benötigt, damit dieser in das für den Matchmaking Algorithmus verständliche Format überführt werden kann.

Die Qualität von Suchergebnissen ist von drei Faktoren abhängig (1) Wie gut die Serviceanbieter ihre Services beschreiben (2) Wie gut ein Such-Request formuliert ist (3) Intelligenz des Service Matchmaking Algorithmus.

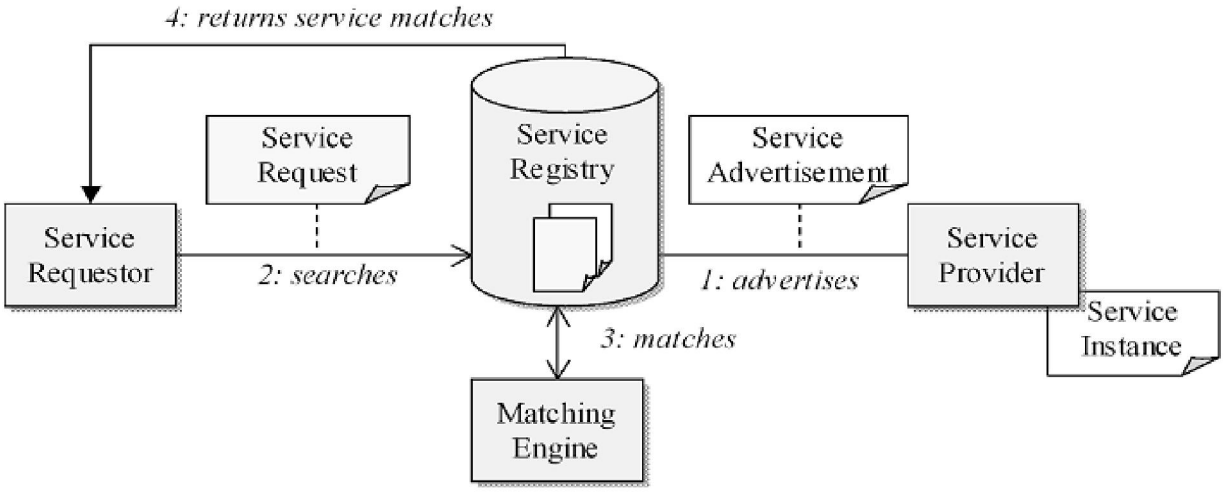

Abbildung 8 Aufbau einer semantische Service Registry (Cardoso, 2007)

In der Praxis gibt es Versuche, die klassische UDDI um semantische Suche zu erweitern. Eine mögliche Erweiterung ist zum Beispiel die Abbildung von semantisch beschriebenen Informationen in die festen UDDI-Datenstrukturen. Ein Beispiel für eine solche Architektur ist in (Srinivasan, et al., 2005) beschrieben. UDDI muss jedoch um einen zusätzlichen Port erweitert werden, damit ein externes semantisches Suchprogramm die Registry durchsuchen kann. Die Suchanfragen werden an dieses Programm gestellt und nicht an die Registry selbst. An die UDDI-Registry können weiterhin gewöhnliche klassische Suchanfragen gestellt werden. Eine UDDI-Registry verfolgt einen zentralen Ansatz, wo die Beschreibungen von registrierten Webservices in einer gemeinsamen Datenbasis (UDDI-Registry) gespeichert sind. Dem Semantic Web liegt jedoch ein dezentrales Konzept zugrunde, so werden die Informationen in einer gewaltigen über das gesamte WWW verteilten semantischen Datenbasis[13] gespeichert. Die semantische Beschreibung eines Webservices ist auch eine Information und kann beispielsweise mit Hilfe von OWL-S erstellt sein. Das OWL-Dokument mit der semantischen Beschreibung eines Webservices kann auf einem Server veröffentlicht werden und somit das Gesamtwissen der semantischen Datenbasis um das in dem Dokument enthaltende Wissen erweitern. So kann ein Webservice-Anbieter seine Webservices semantisch in OWL-S beschreiben und diese OWL-Dokumente auf dem gleichen Server veröffentlichen, wo die Webservices installiert sind. Nun gibt es zwei Möglichkeiten diese Dokumente in der WS-Discovery einzubeziehen: (1) OWL-Dokumente werden in die Registry manuell eingetragen (2) Eine Suchmaschine durchsucht das Internet und findet OWL-Dokumente, dann trägt sie diese in die Registry ein (Jaeger, et al., 2005). Bei einem dezentralen Ansatz einer semantischen Service Registry bleibt die semantische Beschreibung eines Webservices immer aktuell

[13] Semantische Datenbasis ist in Abschnitt 2.3 beschrieben

1.6 OWL-S

„Die Herausforderung im Umfeld der Service-orientierten Architekturen besteht darin, Semantik so eindeutig formulieren zu können, dass menschliche Interpretationsvorgänge verzichtbar werden, ohne Semantik zu verlieren.“

(Dostal, et al., 2004)

Web Ontology Language for Services (OWL-S) ist eine Empfehlung von W3C und wurde für die semantische Beschreibung von Webservices entwickelt (Martin, et al., 2004). OWL-S ermöglicht eine semantische Beschreibung von Webservices und bildet somit die Grundlage für die automatische Webservice Discovery, Ausführung und Komposition (Martin, et al., 2004). Ein Service wird in OWL-S mit Hilfe von ServiceProfile, ServiceModel und ServiceGrounding beschrieben(Abbildung 9):

ServiceProfile

ServiceProfile beschreibt, was ein Webservice macht. Es ist beispielsweise möglich einen Webservice anhand seiner Tätigkeit zu klassifizieren. So wurden in dieser Arbeit im ServiceProfile dem Service ***Kategorien*** hinzugefügt. Dadurch wurde der Service klassifiziert und kann später mit Hilfe von Facetten semantisch gefunden werden (siehe Anhang Abbildung 38).

ServiceModel

ServiceModel beschreibt die Funktionsweise eines Webservices. Zum Beispiel die ***Input-, Output Variablen, Vorbedingungen (Precondition) und Auswirkungen (Effects)***. Bei komplexen Prozessen werden die Prozesse selbst semantisch beschrieben und können später in automatischen Schlussfolgerungen eines ***Agent-Programms*** berücksichtigt werden (siehe Anhang Abbildung 39). Zum Beispiel kann während des Discovery-Prozesses der Abbildung 7 (Seite 19) aus zwei semantisch beschriebenen Prozessen(Prozess-1,-2) ein neues schlussgefolgert werden (Prozess-3).

ServiceGrounding

ServiceGrounding beschreibt die Informationen, die für die Kommunikation mit dem Webservice nötig sind, wie zum Beispiel die Adresse eines Webservices, welches Protokoll genutzt wird (SOAP, REST), wie die Input und Output Variablen in und aus den Nachrichten abgebildet werden. Solche Informationen sind normalerweise in einem WSDL-Dokument enthalten, deswegen ist es relativ einfach, ein Grounding für einen Atomic Prozess aus einem WSDL-Dokument zu generieren (Martin, et al., 2004). So gibt es beispielsweise in dem OWL-S Editor (Plugin für Protegé) (The082) einen WSDL-Importer, der ein Grounding automatisch aus einem WSDL-Dokument generieren kann. Für eine vollständige Definition eines Groundings sind beide Sprachen nötig OWL-S und WSDL. Die Abbildung 41 (siehe Anhang) zeigt, dass OWL-S und WSDL nicht das gleiche Konzept abdecken und nur zum Teil überlappen. WSDL nutzt XML-Schema für die Beschreibung von Input und Output Variablen, wobei OWL-S die Nutzung von OWL-Klassen ermöglicht. WSDL kann das Konzept von OWL-S nicht abdecken weil es nicht möglich ist, die Semantik von OWL-Klassen mit XML-Schema auszudrücken. Genauso kann OWL-S die Binding Informationen nicht semantisch beschreiben und verwendet dafür abstrakte Klassen, die die abstrakte Typen von in WSDL deklarierten ***message parts*** enthalten. So verlässt sich OWL-S auf das WSDL-Binding Konstrukt, die Message für eine Webservice Implementierung korrekt zu formulieren (Martin, et al., 2004). Der Prozess in OWL-S ist nicht

mit einem ausführbaren Programm zu verwechseln, vielmehr ist der Prozess eine semantische Beschreibung eines ausführbaren Programms, wobei Composite Processes durchaus die Geschäftslogik eines Workflow Prozesses enthalten können (Martin, et al., 2004). OWL-S unterscheidet drei Arten von Prozessen, die in dem ServiceModel enthalten sein können: (1) ***Atomic Processes:*** Enthalten keine weiteren Unterprozesse und beschreiben normalerweise eine Operation eines implementierten Webservices. ***(2) Simple Prozesse:*** Abstrakte Prozesse, die einen bestimmten Sachverhalt beschreiben. ***(3) Composite Processes:*** Können weiteren Atomic oder Composite Prozesse enthalten. Enthalten die Geschäftslogik eines Workflow Prozesses.

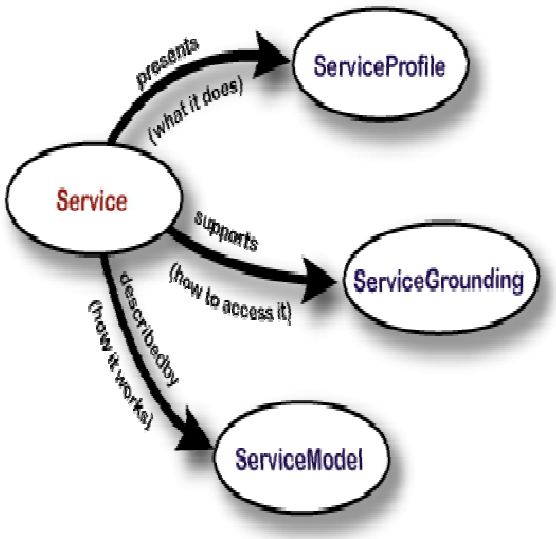

Abbildung 9 Top Level von der Service Ontologie (Martin, et al., 2004)

In der Abbildung 10 ist ein Beispiel abgebildet wie die Webservices mit Hilfe von OWL-S beschrieben werden können. In dieser Arbeit wurde der OWL-S Editor genutzt, das ist ein Plugin für Protegé(The082). Dieser ist speziell an die Nutzung eines OWL-Frameworks angepasst, das für die Beschreibung von Webservices mit Hilfe von OWL-S erstellt wurde. In der Abbildung 10 ist ein OWL-Dokument dargestellt, das fünf Service-Beschreibungen enthält. Diese fünf Services könnten in mehreren OWL-Dokumenten verteilt sein. Jeder Service hat ein Profile, Model und Grounding. Der Service-1 wird durch Profile-1 beschrieben, der die Kategorien enthält, die einen Service klassifizieren und dessen semantische Suche ermöglichen. Prozess-1 beschreibt die Operation-A1 von Webservice-A. Prozess-1 ist ein Atomic Prozess, der den Input, Output, Vorbedingungen und die Auswirkungen bei der Ausführung von Operation-A1 beschreibt. Grounding enthält die Informationen, die gewöhnlich in einem WSDL-Dokument enthalten sind und sind für die Kommunikation mit dem Webservice nötig. So enthält Grounding-1 die Informationen, wie mit dem Webservice-A kommuniziert werden kann und beschreibt, wie eine SOAP Nachricht erstellt werden kann, welche Adresse Webservice-A hat und vieles mehr. Service-2 ist analog zu Service-1 aufgebaut jedoch beschreibt dieser die Operation-A2 des Webservice-A. An dieser Stelle wird es klar, dass ein Atomic-Prozess einer Beschreibung einer Operation eines Webservices entspricht. Service-3 enthält einen komplexen Prozess

(Composite Process). In OWL-S kann ein komplexer Prozess mit Hilfe einer beliebigen Sprache definiert werden. Dieser enthält die Unterprozesse, die wiederum komplexe oder atomare Prozesse sein können. Im OWL-S Editor wird eine Workflow Sprache verwendet, die ähnlich wie WS-BPEL die Webservices orchestriert. Prozess-3 beispielsweise orchestriert drei atomare Prozesse: Prozess-1, Prozess-2 und Prozess-4. Dabei besitzt Service-3 kein Grounding, weil er keinen Service direkt aufrufen kann, sondern nur indirekt über die anderen Prozesse. Service-4 ist analog dem Service-1 und dem Service-2 aufgebaut, jedoch beschreibt dieser eine Operation eines anderen Webservices (Webservice-B), der nicht über SOAP sondern über das REST Protokoll kommuniziert. So enthält Grounding-4 die nötigen Informationen um die Nachrichten für die Operation-B1 von dem Webservice-B zu generieren. Service-5 beschreibt die gleiche Operation wie Service-4 jedoch aus einer anderen Sicht. So kann Service-4 von einem Software-Designer erstellt werden, und der Service-5 kann aus der Sicht eines Datenbank-Spezialisten entwickelt sein. Service-4 und Service-5 bilden den Wissensgehalt über die Operation-B1 und ermöglichen während einer semantischen Suche deren schnelleres Auffinden. Da Service-5 die gleiche Operation wie der Service-4 semantisch beschreibt, verwendet Service-5 einige Informationen aus dem Service-4 wieder. Die Wiederverwendung von Informationen werden im OWL-Standard durch Referenzieren mittels einer URI realisiert. So enthält Profile-5 einige in dem Profile-4 definierten Kategorien-Individuuen. Genauso enthält Service-5 das gleiche Grounding wie Service-4. Da ein Grounding die Kommunikation mit einem Webservice beschreibt und Service-4 und Service-5 die gleiche Operation eines Webservices beschreiben ist auch die Kommunikation der Operation-B1 gleich.

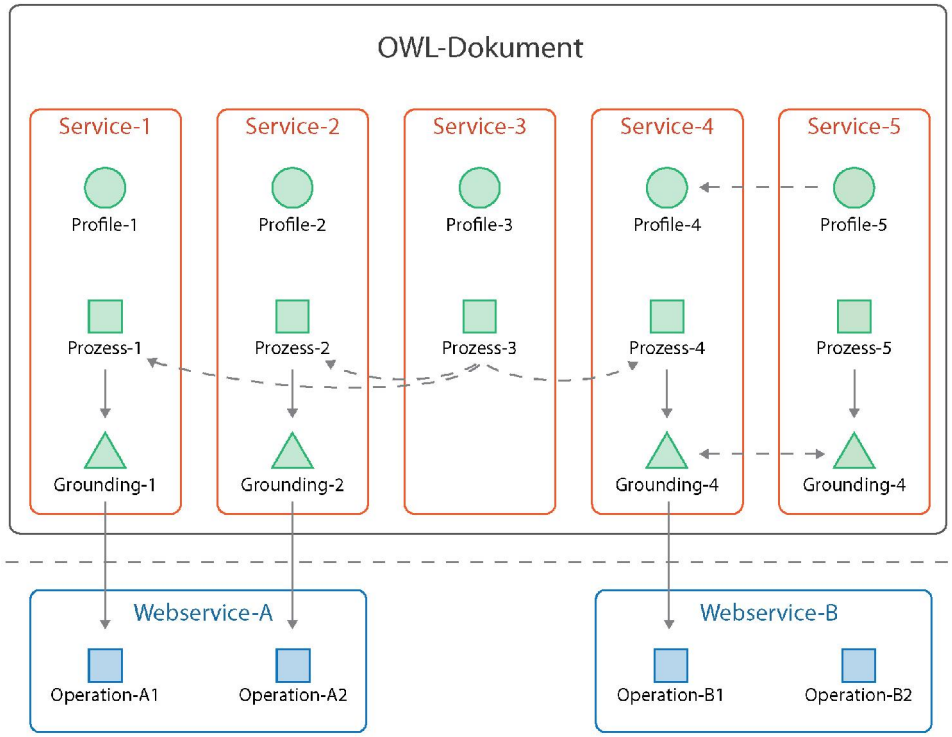

Abbildung 10 OWL-S Beispiel

1.7 Planer

„Die Vision im Umfeld von Semantic Web ist eine Welt, in der sich weiterentwickelnde Service-Ansammlungen und -Vernetzungen so gestaltet sind, dass Agenten und Applikationen miteinander automatisiert kommunizieren können."

(Bauer, 2008)(Seite 149)

Die Webservices sind im WWW verteilt und bieten verschiedenste Funktionalität an. Durch die Orchestrierung von angebotenen Webservices kann eine neue Funktionalität entstehen, die auch als ein Webservice angeboten werden kann und von einer anderen Anwendung ebenso orchestriert werden kann. Die Workflow Prozesse müssen derzeit manuell von einem Menschen erstellt werden. Ein Planer ist ein Programm, das die Orchestrierung der Webservices automatisieren kann. Das Ziel beim Einsatz eines Planer Programms ist die Erreichung eines höheren Automatisierungsgrades in der Erstellung und Ausführung von Prozessen(Rao, 2004). Für die Planung von Webservice Komposition sind semantische Beschreibungen von Webservices unerlässlich. Für die semantische Beschreibung können in OWL-S definierten Ontologien genutzt werden. Wichtige Fragestellungen für ein Planer sind (1) Welches ist der Ausgangs und Zielzustand für die Planung? (2) Nach welchen Kriterien sollen die Webservices gesucht werden? (3) Sind alle Domäneninformationen ersichtlich? (4) Wie soll ein Planer auf unerwartete Ergebnisse eines Webservices reagieren? (Jaeger, et al., 2005). Der Planer soll anhand semantischer Beschreibung eines Webservices erkennen ob dieser für die Planung relevant ist und falls ja dann die Möglichkeit haben einen Webservice anhand semantischer Beschreibung zur Laufzeit einzubinden. Für die automatische Orchestrierung von Semantic Webservices müssen folgende Kriterien erfüllt sein: ***(1) Deklaration von Synonymen:*** identische Inhalte mit unterschiedlichen Namen sollen untereinander verknüpft sein. ***(2) Reasoning:*** das logische Erschließen von neuen Zusammenhängen. ***(3) WS-Discovery/Selection:*** die für die Erstellung nötigten Dienste für einen neuen Dienst müssen gefunden und für die Orchestrierung bereitgestellt werden (Dostal, et al., 2004). Die Kompositionsplanung kann nach (Rao, 2004) in mehre Phasen gegliedert werden.

1. **Präsentation eines Services**: Die Serviceanbieter veröffentlichen ihre Services auf dem Markt. Dabei können sie die einzelnen Operationen eines Webservices beschreiben (Atomic Service), oder aber die Workflow Prozesse definieren, die andere Services orchestrieren (Composite Service).
2. **Sprachübersetzung:** Die meisten Kompositionsmethoden unterscheiden zwischen externen und internen Sprachspezifikationen. Die externen Sprachen verbessern die Kommunikation mit dem System und die internen die Komposition des eigentlichen Prozesses.
3. **Generierung eines Models für den Kompositionsprozess:** Inzwischen kann ein Service Requester seine Anforderungen in einer speziellen Sprache formulieren. Der Prozessgenerator nimmt die gestellten Anforderungen als Input und versucht diese zu lösen, indem er einen Prozess orchestriert, der den gestellten Anforderungen entspricht.
4. **Evaluation von dem erstellten Prozess:** Der Prozess-Generator kann mehrere Prozesse erstellen und diese dann nach in der Anforderung enthaltenen Bedingungen bewerten. Dabei ist der Prozess mit der höchsten Bewertung auch der gelungenste.
5. **Ausführung eines erstellten Prozesses:** Nachdem ein Prozess ausgewählt wurde, kann dieser ausgeführt werden.

In Workflow basierten Prozessen kann ein Prozess auf zwei verschiedenen Arten erstellt werden: (**statisch)** Der abstrakte Prozess ist in der Requester-Anfrage enthalten. In diesem abstrakten Prozess ist der erwartete Workflow Prozess vordefiniert. Der Planer muss die abstrakten Services dieses Prozesses gegen die richtigen ersetzen. Dazu sucht der Planer nach Webservices, die den Anforderungen des abstrakten Prozesses genügen, vergibt den gefundenen Webservices nach einem bestimmten Vorgehen die Priorität und ersetzt die abstrakten Services in dem abstrakten Prozess mit gefundenen Webservices mit höchster Priorität. (**dynamisch)** Die meisten dynamischen Methoden sind stark mit KI Methoden verwandt. Die Voraussetzung für solche Methoden ist, dass für die Prozesse eine semantische Beschreibung der Vorbedingung sowie der Effekte existiert. Die Vorbedingungen eines Services können mit dem aktuellen Zustand der Welt verglichen werden. Falls der Weltzustand die Vorbedingungen erfüllt, kann der Service ausgeführt werden. Unter dem Weltzustand werden dabei der Speicherzustand des zu planenden Prozesses, sowie die Beschränkungen von Input und Output Variablen der Suchanfrage verstanden. Ein Effekt eines Services beschreibt, wie der Weltzustand nach dessen Ausführung verändert wird. Dadurch kann der Planer die Services auswählen, die den Weltzustand in der geeigneten Art und Weise verändern. Ein Prozess wird automatisch durch die Anwendung eines logischen Theorems erstellt, ohne dass ein abstrakter Prozess wie bei der statischen Methode vordefiniert sein muss (Rao, 2004) (Seite 22). Neben zuvor beschriebenen Kompositionsmethoden kommen auch die Methoden der künstlichen Intelligenz für automatische Komposition oft zum Einsatz. Da OWL-S die Vorbedingungen und Effekte eines Services beschreiben kann, existiert aktuell eine direkte Anbindung von OWL-S an einige Methoden der künstlichen Intelligenz (Rao, 2004) (Seite 23). Verschiedene Kompositionsmethoden bieten verschiedene Level der Automatisierung. Die Orchestrierung von Webservices ist eine sehr komplexe Aufgabe, deswegen ist es derzeit nicht möglich alles automatisch zu generieren. Automatische Orchestrierung kann jedoch viele Aufgaben automatisch transparent für den Entwickler erledigen und somit das Grundgerüst für einen Workflow Prozess erstellen. Dieses Grundgerüst kann später durch den Entwickler vollendet werden. Eine ausführliche Diskussion über verschiedene Automatisierungsmethoden ist in (Hull, et al., 2003) gegeben.

1.8 Matchmaking Algorithmen

Die semantischen Matchmaking Algorithmen können nach verschiedenen Kriterien klassifiziert werden, so beispielsweise werden die Matchmaking Algorithmen in (Cardoso, 2007) nach zwei Kategorien unterschieden: ***(1) direkt:*** Ein Request wird gegen einzelne Service Advertisements verglichen, ***(2) indirekt:*** Ein Request wird gegen einen Composite Service verglichen, der aus anderen Composite Services oder einfachen Services besteht. Es gibt verschiedene indirekte Matchmaking Algorithmen. ***Semantic Capabilities Matching*** nutzt beispielsweise nur die ServiceProfile für die Suche nach gewünschten Webservices, alle anderen in der semantischen Beschreibung eines Webservices enthaltenen Informationen werden ignoriert. Eine andere Suchmethode ist ***Multilevel Matching,*** diese nutzt möglichst viele Informationen aus der semantischen Beschreibung eines Webservices. Bestimmte Informationen werden in einer bestimmten Schicht gesucht und bewertet. Jede Schicht bekommt das Gewicht und die gesamte Bewertung entsteht aus der Summe der Bewertungen einzelner Schichten gewichtet mit deren Gewichten. Die Schichten werden dabei für einen in der Suche interessanten Aspekt zusammengestellt. So kann eine Schicht eine Logik implementieren, die den Webservice nach der Übereinstimmung mit dem in dieser Schicht festgelegten Suchkriteriums bewertet. Eine weitere Suchmethode ***(DL Matchmaking with Service Profile Ontologies)*** geht eher Richtung

der Suche durch das Schlussfolgern. In dieser Suchmethode sollten die gesuchten Webservices mit Hilfe von Description Logic(DL) beschrieben werden. Dabei wird die semantische Beschreibung eines Webservices in dem ServiceProfile gespeichert, alle anderen Teile des OWL-Standards wie Prozessmodell oder Grounding werden in der Suche nicht benötigt. Die Suche eines Webservices geschieht durch das Schlussfolgern einer Inferenzmaschine. Eine Erweiterung der gerade beschriebenen Suchmethode sind die ***(Similarity Measures and Information Retrieval Techniques).*** Sie greift auf die klassische syntaxbasierte Webservice Suche zurück, falls die Suche mit Hilfe von Schlussfolgern fehlgeschlagen ist. Zum Beispiel könnte eine Textsuche nach festgelegten Schlüsselwörtern in einer semantischen Beschreibung durchgeführt werden. Einen vollkommen anderen Ansatz bietet die Suche nach einem bestimmten Muster in der semantischen Beschreibung ***(A Graph-Based Approach)***. Die Mustersuche erfolgt über den Graphen, der die semantische Beschreibung eines Webservices repräsentiert (zum Beispiel RDF-Graph). In dem erzeugten Graph wird nach einem Muster gesucht und alle Webservices zurückgegeben die diesem Muster entsprechen. Es ist möglich nach einem festen Muster mit einer Abfragesprache wie SPRAQL zu suchen. Der Nachteil dieser Suche ist, dass sie den Grad der Übereinstimmung (unscharfe Suche) nicht unterstützt, d.h. ein Muster wird entweder gefunden oder nicht. Jedoch ist diese Methode recht einfach zu implementieren und für prototypische Implementierung ausreichend. In dieser Arbeit wurde dieser Ansatz daher für die Suche von Webservices, mathematischen Gleichungen, Dokumenten und anderen Ressourcen verwendet. Eine weitere Suchmethode ist für die automatische Komposition von Workflow Prozessen geeignet. Die ***Indirect Graph-Based Matching*** Suche nutzt die semantische Beschreibung von Webservices für die Generierung neuer Prozesse. Das ist möglich da die semantische Beschreibung eines Webservices auch Composite Prozesse enthalten kann. Ein Prozess beschreibt die Funktionsweise eines Semantic Services, sowie seine mögliche Nutzung. Die automatische Komposition eines Prozesses erfolgt mit Hilfe eines Graphen, der alle Prozesse enthält und somit ihre Überschneidungen veranschaulicht. Der Suchalgorithmus versucht die Knoten dieses Graphen so miteinander zu kombinieren, dass die Inputs und Outputs aus der Suchanforderung erfüllt werden. Beim Kombinieren von Knoten aus verschiedene Graphen können neue Prozessketten gebildet werden, die einen Composite Prozess bilden, der die Suchanforderungen erfüllt. Diese Vorgehensweise ist in Abschnitt 1.4 eingehend beschrieben. Eine ähnliche Suchmethode bietet der ***Indirect Backward Chaining Matching*** Ansatz. Dieser ähnelt stark dem Indirect Graph-Based Matching, unterscheidet sich jedoch in der Vorgehensweise. Die Suche erfolgt durch Backtracing und beginnt daher bei der in der Suchanfrage spezifizierten Output-Schnittstelle des geforderten Webserives. Wenn so ein Webservice gefunden wurde, wird durch Rückwärtssuche versucht, eine mögliche Prozesskette von Webservices zu erstellen, damit auch die in der Suchanfrage festgelegten Anforderungen für Input erfüllt werden.

1.9 Facettenbasierte Klassifikation

Klassifikation gehört zu den ältesten Dokumentationssprachen. Viele auch heute noch verwendete Klassifikationen entstanden vor etwa hundert Jahren (DDC[14], DK[15]). Im Laufe der Zeit versuchte man die existierenden Klassifikationen ständig zu verbessern. Dies geschah durch die Anpassung von Klassifikationen an die neuen Entwicklungen in der Technik und

[14] DDC - Dewey Decimal Classification

[15] DK - Internationale Dezimalklassifikation

Wissenschaft, Erhöhung der Ausdrucksfähigkeit der Klassifikationssprache, sowie die Verwendung von besseren Instrumenten für die Erschließung der Inhalte (Manecke, 2004). Eine ***Klassifikation*** ist eine Gruppierung oder Einteilung des gesamten Wissens nach einheitlichen methodischen Prinzipien (Mias, 2000). Eine gelungene Klassifikation von Wissen kommt der Denkweise des Nutzers entgegen (Nohr, 1996). Nach (Manecke, 2004) versucht ein Nutzer die gleichen Dinge zu gruppieren: „Gleiches sollte zu Gleichem". Deswegen haben viele Webdesigner, wahrscheinlich sogar unabhängig voneinander die Facetten für die Gestaltung ihrer Webseiten neu entdeckt (Denton, 2007). Zum Einen lag dies sicherlich daran, dass die international bedeutendsten Systeme erst relativ spät digitalisiert wurden, zum Anderen in der Mentalität der WWW-Gemeinschaft. Denn seit dem Entstehen des WWW dessen treibende Kräfte die alten bewehrten Dinge ablehnten und „das Rad neu erfinden wollten" (Oberhauser, 2004).

Der Begriff „Klassifikation" kann nach seiner Verwendung in drei verschiedene Arten gegliedert werden (Manecke, 2004): (1) „dem Prozess der Klassifikationserarbeitung (d.h. der Klassenbildung);" (2) „dem Klassifikationssystem als Ergebnis des Klassenbildungsprozesses;" (3) „dem Prozess des Klassierens bzw. des Klassifizierens, d.h. dem gegenseitigen Zuordnen von Objekten und Klassen des Klassifikationssystems.". Das Zuordnen von Objekten erfolgt auf der Grundlage mindestens eines gemeinsamen ***klassifikatiorischen Merkmals (Klassem).*** Ein Klassem gehört zu einer Klasse und es unterscheidet diese von anderen Klassen. Hinter einer ***Klasse*** verbirgt sich ein dreistufiger Abstraktionsprozess (Manecke, 2004): (1) Abstraktion des Sachverhalts zu einem Begriff, der die Merkmale bestimmt, die diese Klasse von einer anderen unterscheidet, (2) Ausdrücken des Begriffs durch eine äquivalente Bezeichnung, (3) Einordnung einer Klasse in die Suchhierarchie(Über-/Unterordnung) und die Definition von systemabbildenden und von natürlichen Sprachen unabhängigen Bezeichnungen (Notationen). Eine ***Notation*** kann beispielsweise eine nach bestimmten Regeln gebildete Zeichenfolge sein, die eine Klasse repräsentiert und deren Stellung im systematischen Zusammenhang abbildet (Manecke, 2004). Die Klassen können in ***Basisklassen*** (engl. basic classes) und in ***komplexe Klassen*** (engl. compound classes) unterteilt werden (Nohr, 1996). In den Basisklassen werden die atomaren Wissenseinheiten präsentiert. Zum Beispiel werden in dieser Arbeit Webservices mit Hilfe von OWL-S Kategorien klassifiziert. So eine Kategorie entspricht einer atomaren Wissenseinheit und wird durch eine Basisklasse repräsentiert. Eine komplexe Klasse ermöglicht die Definition einer Hierarchie, dabei können die Klassen untergeordnete und übergeordnete Klassen haben (Mias, 2000). Zum Beispiel in der Abbildung 11 ist die Klasse Hut eine komplexe Klasse, und sie enthält alle Objekte die der Klasse Damenhut und der Klasse Herrenhut zugeordnet sind.

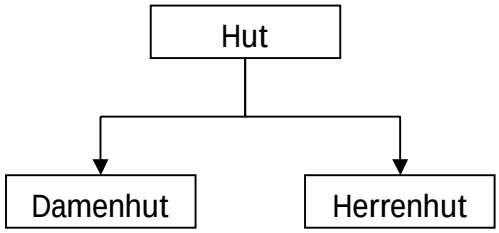

Abbildung 11 komplexe und Basisklasse (Beispiel aus (Mias, 2000)).

Die Beziehung zwischen einer komplexen und einer Basisklasse entspricht der Beziehung zwischen einem ***Gattungsbegriff*** und einem ***Artbegriff***. In der Abbildung 12 ist der

Zusammenhang zwischen einem Gattungsbegriff und einem Artbegriff an einem Schiff Beispiel veranschaulicht. Das Schiff ist der Gattungsbegriff und dieser wird in weitere Artbegriffe unterteilt: Fahrgastschiff, Frachtschiff, Fischerschiff. Ein Artbegriff der weiter unterteilt werden kann ist ebenso ein Gattungsbegriff. So ist der Frachtschiff aus der Sicht von Schiff ein Artbegriff, aber da dieser Begriff weiter in Tankschiff, Kühlschiff, Massengutschiff unterteilt wird, auch ein Gattungsbegriff (Abbildung 12).

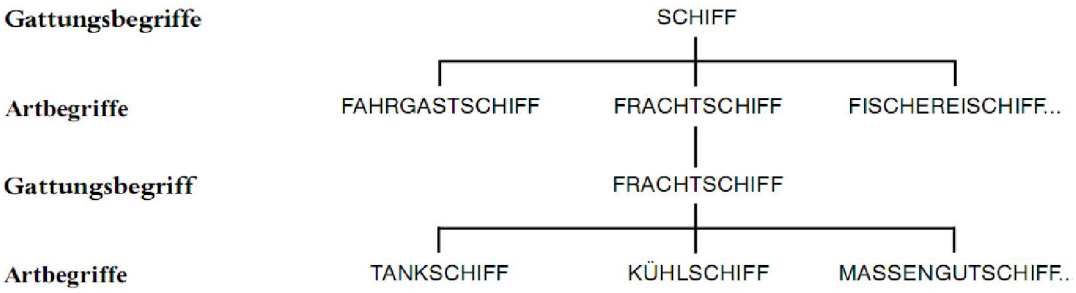

Abbildung 12 Zusammenhang zwischen einem Art- und einem Gattungsbegriff (Manecke, 2004)

Für die Klassifikationen gibt es zwei Arten von Hierarchien (1) ***Starke Hierarchie*:** Jeder Artbegriff hat nur einen Gattungsbegriff, es entsteht eine Art Begriffspyramide und (2) ***Schwache Hierarchie*:** Ein Artbegriff kann mehrere Gattungsbegriffe haben. Die starke Hierarchie ermöglicht die Suche nur nach einem Suchkriterium (= eindimensionale Suche), wobei die schwache Hierarchie die Suche unter mehreren Suchkriterien (mehrdimensionale Suche) möglich macht (Manecke, 2004). Zukünftig „werden die flexiblen Strukturen einer schwachen Hierarchie mehr genutzt werden, d.h. man geht von einer eindimensionalen Recherche hin zur mehrdimensionalen Suche.“ (Mias, 2000).

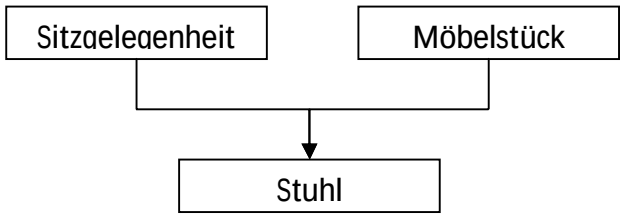

Abbildung 13 Eine schwache Hierarchie(Beispiel aus (Mias, 2000)).

Bei der Definitionen von Klassen sollen die drei Hauptpunkte beachtet werden (Seite 128 (Manecke, 2004)): (1) Der Begriffsumfang des Gattungsbegriffs muss den Umfang der Artbegriffe einschließen, (2) die Artbegriffe müssen disjunkt sein, (3) es darf nicht gleichzeitig nach verschiedenen Merkmalen gegliedert werden. So beispielsweise ist eine „Unterteilung der Klasse Schuhe in der nächsten Ebene nach Damenschuhe, Herrenschuhe, Sommerschuhe und Winterschuhe nicht möglich, da hier einmal nach Person und einmal nach der Jahreszeit in einer Ebene unterschieden wird“ (Mias, 2000). In Bibliotheken ist es fast nie möglich die oben aufgelisteten drei Kriterien einzuhalten, deswegen muss oft ein Kompromiss getroffen werden. Zum Beispiel wenn ein Damensommerstoffschuh eingeordnet werden soll, würde das zu einem Problem in einer starken Hierarchie führen denn wo soll dieser Schuh eingeordnet werden in

die Damenschuhe oder in die Sommerschuhe. In (Manecke, 2004) werden zwei Arten von Klassifikationen unterschieden (Seite 130):

- ***analytische:*** Ist eine starr strukturierte analytische Klassifikation. Hier werden die im während der Systematik zusammengeführten Begriffe vom Allgemeinen zum Speziellen immer feiner untergliedert.
- ***analytisch-synthetische:*** Die gleichrangige Merkmalsbegriffe bilden die Begriffsgruppen auch Kategorien oder Facetten genannt. In diese Gruppen können nun die Einzelbegriffe (auch Foci oder Isolate genannt) einsortiert werden. Die notwendige Untergliederung erfolgt durch weitere Facetten (Unterfacetten). Die analytisch-synthetische Klassifikation ist auch als ***Facettenklassifikation*** bekannt. Die Facettenklassifikation ist in der Regel ***hierarchisch*** und ***mehrdimensional***.

In der Praxis werden die Prinzipien von beiden Klassifikationstypen miteinander vermischt, um die Vorteile aus beiden Klassifikationsarten zu vereinen. Die Auswahl von richtigen Facetten ist entscheidend und erfordert das Fachwissen von zu klassifizierenden Objekten. Die Facettenbasierte Klassifikation beschreibt umso detailierter das Wissen, je kleiner der zu beschreibende Bereich ist. Einige Beispiele für Design von Facettenklassifikation sind in (Nielsen, 2000) aufgeführt. Die drei Hauptprobleme der Facettenklassifikation sind: (1) die Schwierigkeit die richtigen Facetten zu wählen, (2) die Schwierigkeit die Beziehungen zwischen Facetten auszudrücken, (3) die Schwierigkeit die Facetten und ihre Beziehungen zu visualisieren. Nach (Denton, 2007) gibt es zwei grundlegende Möglichkeiten die Facettenklassifikation nutzbar zu machen:

- **Stichwortsuche:** Der Benutzer gibt ein oder zwei Wörter und das System durchsucht die Facetten und Entitätsbeschreibungen und zeigt die Ergebnisse an. Es ist schwierig viele Ergebnisse anzuzeigen, weil nicht klar ist welche Facetten nun für den Nutzer wichtig sind, damit dieser seine Suche verfeinern kann.
- **Facettenbasierte Navigation:** Der Nutzer formuliert die Anfrage an das System in dem er beispielsweise die Facetten anklickt und dadurch seine Anfrage immer mehr verfeinert. Diese Art der Navigation bietet eine Möglichkeit komplexe Anfragen an das System zu formulieren ohne diese Anfragen wie gewöhnlich per Hand schreiben zu müssen (Polowinski, 2008), zum Beispiel in SPARQL.

Einer der Vorteile der Facettenklassifikation liegt daran, dass die Facetten so definiert werden können, dass der Suchpfad immer zu Resultaten führt und nicht wie bei gewöhnlicher Suche keine Ergebnisse liefert (Polowinski, 2008). Der Suchpfad kann mit Hilfe von einer zuvor definierten Notation erstellt werden und beschreibt den Ort, wo die gefundenen Ressourcen gespeichert sind (Manecke, 2004). Manchmal ist es nötig, einem Nutzer eine Möglichkeit zu geben die Ergebnisse in einem Bereich einzugrenzen, wie beispielsweise in einer Zeitspanne oder in einem Preisbereich (Polowinski, 2008). Diese Art von Suche trifft man oft bei E-Commerce wie EBay oder Amazon. Oft ist es auch notwendig mehrere Werte in einer Facette auswählen zu können. Am besten lassen sich mehrere Werte in einer Facette durch die Verwendung von Checkboxen und Bäumen auszuwählen (Polowinski, 2008). Interessant ist in diesem Zusammenhang auch die Aussage aus (Denton, 2007), dass Facetten nur dann eingesetzt werden sollten, wenn Hierarchien, Bäume, Paradigmen für die Suche von Informationen nicht geeignet sind.

2. **Synthese**

In diesem Kapitel werden die während dieser Arbeit erarbeiteten Konzepte zur Anpassung von semantischer Suche sowie die Anpassung einer Workflow Entwicklungsumgebung an die Bedürfnisse des Bauwesens vorgestellt. Zuerst wird die Problematik der Bauüberwachung eines Bauwerks sowie die gestellten Anforderungen zu deren Lösung anhand eines Beispiels verdeutlicht. Danach folgt die Erläuterung des an die semantische Suche angepassten Konzeptes zur Verwendung von Informationen aus der semantischen Datenbasis. Danach folgt ein Abschnitt über die semantische Datenbasis. Dabei wird erläutert, was die semantische Datenbasis ist und wie sich diese von einer gewöhnlichen relationalen Datenbank unterscheidet. Darauffolgend wird das Konzept für eine generische Verwendung von Ingenieurmodellen erläutert. Am Ende dieses Kapitels werden die während dieser Arbeit erarbeiteten Techniken für die Anwendung zur Facettenbasierten Klassifikation vorgestellt.

2.1 Problematik im Bauwesen

„Die tragischen Einstürze von Hallendächern mit mehreren Todesfällen im Winter 2006 haben große öffentliche Diskussionen über die Zuverlässigkeit von Tragwerken ausgelöst. Es bleibt zu hoffen, daß dies zu einer Erhöhung des Bewußtseins für die Notwendigkeit der Überwachung der Systemzustände und der rechnerischen Erfassung des tatsächlichen Verhaltens und nicht nur eines Bemessungsnachweises von Ingenieurbauwerken führt."

(Scherer, et al., 2007)

Die Ingenieurmethoden zur **Bauwerküberwachung** *und –berechnung sind bereits auf einem fortschrittlichen Stand. Durch diese Ingenieurmethoden kann der Zustand von Tragwerken eingeschätzt, sowie die notwendigen Instandhaltungsmaßnamen abgeleitet werden. Die lückenlose Inspektion einer großen Anzahl von Bauwerken ist sowohl aus Zeitgründen als auch aus Kostengründen unmöglich. Durch die Installation der Sensorsysteme könnten viele Daten automatisch aufgezeichnet werden. Die Auswertung dieser Daten könnte für die Einschätzung des Zustandes der Tragwerke sowie der Ableitung von notwendigen Instandhaltungsmaßnamen genutzt werden. Die Hauptaufgabe der Bauwerküberwachung ist es anhand von aufgezeichneter Messdaten zu prüfen, ob* „(1) das Verhalten des Systems innerhalb vorgegebener Grenzen liegt und (2) die Modellannahmen bei der Planung des Systems richtig waren und nach wie vor aktuell sind" *(Faschingbauer, 2007). Deswegen sollte während der Bauwerksüberwachung die Vereinbarung der Beobachtung und des Modells geprüft werden. Bauwerksüberwachung hängt stark von der Planung und Bauausführung ab. Der Gesamtprozess ist zyklisch und kann in Vorhersage, Beobachtung, Interpretation und Ausführung von Maßnahmen unterteilt werden (Abbildung 14). Die* **Vorhersage** *ist eine iterative Aufgabe zur Findung einer optimalen Lösung für einen Bauwerk. Bei der Bauwerksüberwachung bildet die Vorhersage die Basis für die Auswahl und Platzierung von Sensoren, sowie die Festlegung der Messzeitpunkte. Damit das Vorhersagemodell an die Sensordaten angeglichen werden kann, muss die Vorhersage mehrmals während der Bauzeit wiederholt und aktualisiert werden. Heutige Technik bietet viele Sensoren für viele Messgrößen die die zerstörungsfreie dauerhafte* **Beobachtung** *von Bauwerken ermöglichen.* „Es ist davon auszugehen, dass sowohl die Sensornetze selbst, als auch deren Einsatzhäufigkeit in naher Zukunft erheblich wachsen werden. In gleichem Maße wächst damit aber auch die Menge an Daten, die erfasst, übermittelt, interpretiert und dargestellt werden müssen." *(Scherer, et al., 2007). Das Problem das dabei entsteht, ist die nicht zureichende Datenverarbeitung. Damit die Daten effizient verarbeitet werden können, müssen diese in einer für eine Maschine verständlichen Form gespeichert werden, zum Beispiel im OWL-Format. Solche Daten können leicht wiedergefunden werden. Außerdem kann eine Maschine aus solchen Daten viele Informationen automatisch schlussfolgern. Während der* **Interpretation** *der Messdaten werden die Messwerte mit den Sollwerten aus der Vorhersage verglichen. Außerdem können durch die Versuchsberechnung die Ursachen für die Abweichungen zwischen Mess- und Sollwerten identifiziert werden. Die Resultate der Interpretation werden in* **Präventiven und korrektiven Maßnahmen** *für die Verbesserung des Zustandes des Bauwerkes genutzt. Oft ergeben die Maßnahmen neue Randbedingungen und erfordern damit die Anpassung oder sogar den Austausch des Vorhersagemodells.*

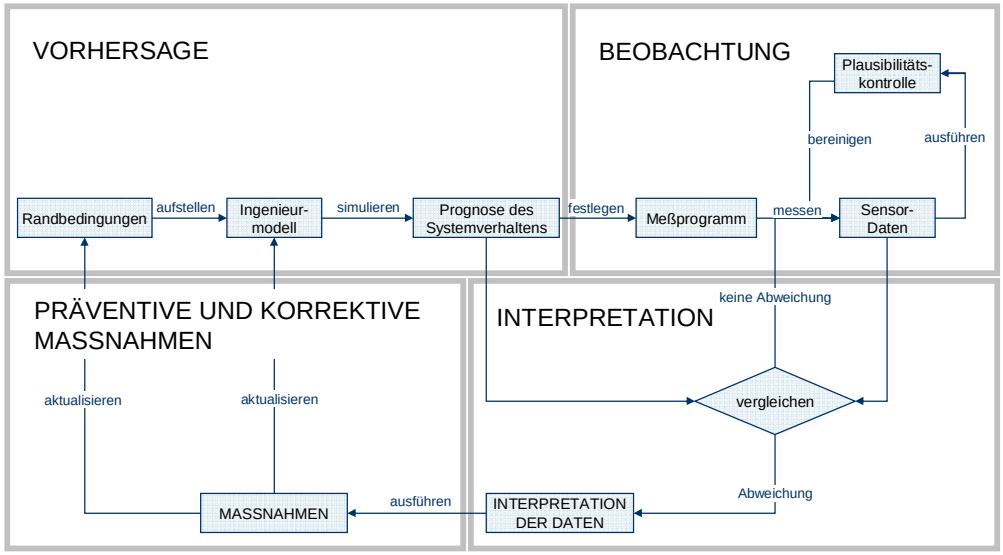

Abbildung 14 Prozessablauf bei der Bauwerksbeobachtung (Faschingbauer, 2007)

Anhand eines praktischen Beispiels aus dem geotechnischen Ingenieurbau soll nun veranschaulicht werden, dass sich für unterschiedliche Phasen des Bauprozesses unterschiedliche Ingenieurmodelle ergeben können. In der Abbildung 15 ist eine schematische Darstellung des Aushubs einer Baugrube veranschaulicht. Die gestrichelte Linie stellt die vorhergesagte Verformung dar, wobei die Punktlinie die durch die Sensoren aufgezeichneten Verformungen veranschaulicht. Nach jeder Messung kann die zuvor gemachte Vorhersage an die neuen Messwerte angepasst werden. In Situation 1 weisen die vorhergesagten und die gemessenen Horizontalverschiebungen nur eine geringe Abweichung auf. Somit kann mit dem gleichen Modell durch die Anpassung der Modellparameter an die Messwerte eine neue Vorhersage gemacht werden. In Abhängigkeit von der Anzahl der zu untersuchenden Parameter können mehrere Modellversionen entstehen (Faschingbauer, 2007). Manchmal treten auch Verformungen auf, die das gerade verwendete Model nicht berücksichtigen kann. Das verwendete Model ist dann nicht mehr für die Vorhersage geeignet und sollte gegen ein Modell, das die auftretenden physikalischen Phänomene repräsentieren kann, ausgetauscht werden. Ein Beispiel dafür bietet die Situation 2, hier ist zusätzlich zur prognostizierten Horizontalverschiebung eine vertikale Setzung eigetreten. Das verwendete Model in der Situation 1 kann jedoch die vertikale Setzung in der Berechnung nicht berücksichtigen. Das heißt, das Modell kann nicht durch die Variation der Parameter an die Messwerte angepasst werden. Damit die Vorhersage in der Situation 2 richtig gemacht werden kann, muss das Modell ausgetauscht werden.

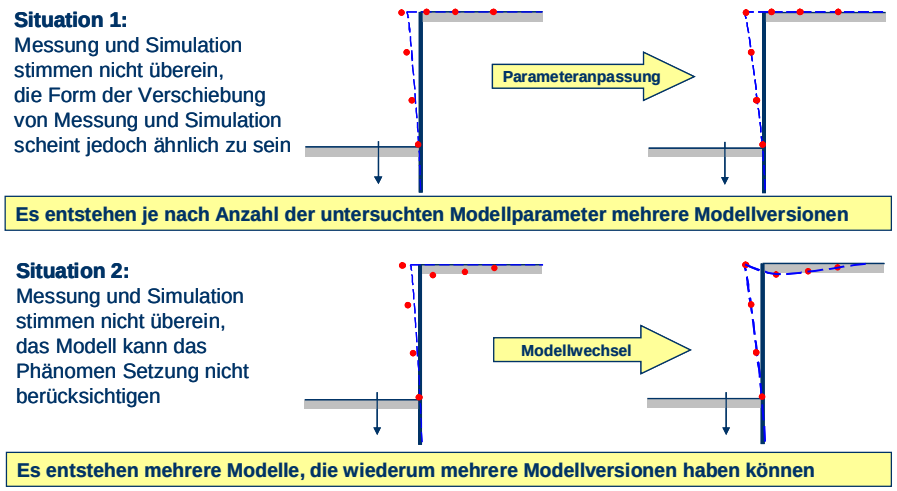

Abbildung 15 Modellanpassung und Modellwechsel bei der Systemidentifikation (Faschingbauer, 2007)

In der Abbildung 16 ist das Vorgehensmodell für die Erstellung eines, an die Lösung des zuvor beschriebenen Problems, angepassten Workflow Prozesses dargestellt. Zuerst werden die benötigten Webservices und Ingenieurmodelle gesucht. Ein Ingenieurmodell ist zum Beispiel das Berechnungsmodell für die Horizontalverschiebung. Während dieser Arbeit wurde ein Webservice entwickelt, der mathematischen Gleichungen berechnen kann und dadurch die Mathematik als eigenständigen Aspekt von der Webservicelogik entkoppelt. Die Funktionsweise dieses Webservices ist in Abschnitt 3.4 eingehend beschrieben. Diese prototypische Implementierung zeigt deutlich dass es möglich ist die Berechnungsmodelle (mathematische Gleichungen) von der Logik zu trennen. Der Vorteil dieses Konzeptes liegt darin, dass die Berechnungsmodelle generisch in einem Berechnungsservice ausgetauscht werden können. Das führt zu einer höheren Flexibilität und ermöglicht es zum Beispiel durch die Verwendung von OSPP einen Workflow Prozess an die neuen Bedingungen schnell anzupassen. Außerdem können die Berechnungsmodelle semantisch beschrieben und dadurch leicht gefunden werden. Ein Workflow Prozess könnte die Businesslogik für die Bauüberwachung von der in der Abbildung 15 dargestellten Baugrube implementieren. Damit dieser erstellt werden kann, müssen die Webservices und dazugehörigen Berechnungsmodelle gesucht werden. Bei einer großen Anzahl von Webservices und Berechnungsmodellen sollte eine semantische Suche angewandt werden. In dieser Arbeit wurde eine semantische facettenbasierte Suche von Webservices prototypisch in OSPP implementiert. Nachdem der geforderte Prozess erstellt wurde, müssen dafür benötigte Daten gesucht werden. Das könnte zum Beispiel das statische Modell einer Baugrube sein, deren Daten für die Berechnung der Horizontalverschiebung verwendet werden können. Die Daten sollten ebenfalls semantisch beschrieben werden, damit diese später mit Hilfe von semantischer Suche gefunden werden können. Nachdem die benötigten Daten gefunden wurden, müssen diese zugewiesen werden und danach kann der Prozess ausgeführt werden. Nach jeder Messung sollte der Bauingenieur die Daten der Baugrube aktualisieren und das weitere Vorgehen neu berechnen. Falls die Situation 2 eintritt, muss der Bauingenieur zum Schritt *WS/Ingenieurmodell suchen* zurückkehren und das alte Berechnungsmodell gegen ein neues, das auch die Setzung berücksichtigt, austauschen. Der Workflow Prozess einer Bauüberwachung muss immer wieder in einer Schleife ausgeführt werden. Nach jeder Iteration muss dieser geprüft und angepasst werden. Bei der Erstellung eines klassischen BPEL-Prozesses führt diese Vorgehensweise wegen Deployment zu einem hohen Aufwand. Diese überflüssige Arbeit kann einem Ingenieur durch das neuartige Konzept der OSPP Entwicklungsumgebung erspart werden. OSPP ermöglicht die

Veränderung eines Workflow Prozesses zur Laufzeit und macht somit den Aufwand überflüssig dieses nach jeder Veränderung neu zu deployen. Der Überwachungsprozess läuft in einer Schleife ab und hält nach jeder Iteration an. An dieser Stelle können die Messwerte mit Sollwerten verglichen werden, die Daten und gegeben falls auch das Berechnungsmodell können an dieser Stelle angepasst werden. Danach kann die Ausführung des veränderten Prozesses fortgesetzt werden. Im Vergleich zum klassischen BPEL-Entwicklungsprozess bietet diese Möglichkeit höhere Flexibilität und führt zu Zeitersparnis.

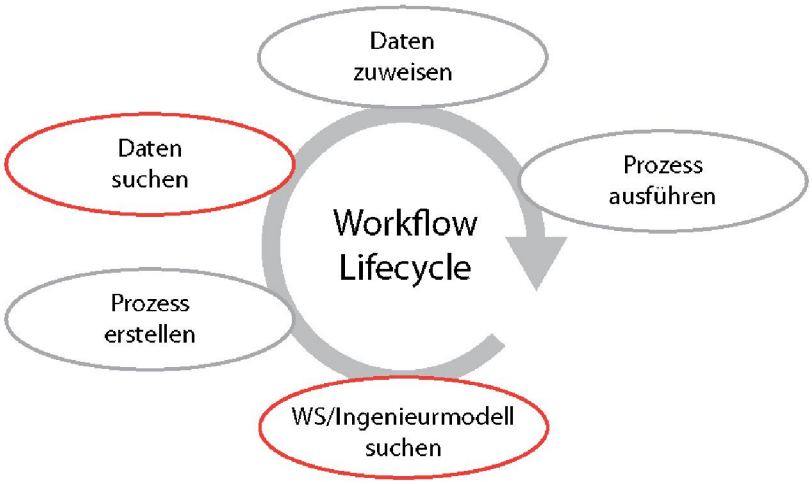

Abbildung 16 Vorgehensmodell

2.2 Konzept zur Integration von Daten, Modellen und Services

Das dieser Arbeit zugrunde liegende Konzept zur Zuordnung von Modellen und Daten zu den Auswertewerkzeugen und deren Einbettung in einen Workflow ist in (Faschingbauer, et al., 2007) beschrieben. In der hier voliegenden Arbeit wurde das ursprüngliche Konzept für die Anwendungen im Semantic Web angepasst, indem die relationale Datenbank gegen die semantische Datenbasis ersetzt wurde. Das ursprüngliche Konzept ist im Anhang in der Abbildung 42 dargestellt. Das für diese Arbeit angepasste ***Konzept*** ist in der Abbildung 17 dargestellt, es beschreibt wie die Informationen aus der semantischen Datenbasis in einem Workflow Prozess (WS-BPEL, OSPP) bereitgestellt und danach verwendet werden können. Da BPEL alle Informationen basierend auf XML-Technologien verarbeitet, müssen alle benötigten Informationen aus der semantischen Datenbasis in XML überführt werden. Auch für die Verarbeitung in OSPP ist die Abbildung der Informationen aus der semantischen Datenbasis in XML notwendig. Die Überführung geschieht mit Hilfe von Meta-Modellen. Dabei werden die OWL-Klassen in die XML-Schema Typen abgebildet. In der Abbildung 18 ist das Konzept der *Projektion* einer OWL-Klasse in einen XML-Schema Typ in Zusammenhang zu OMG-Vierschichten Architektur (Gepting) (OMG) veranschaulicht. Die M0-Schicht enthält OWL-Individuen aus der semantischen Datenbasis und XML-Schema Typ Instanzen eines BPEL-Prozesses. Die in OWL-Individuen enthaltenen Informationen werden mit Hilfe einer festdefinierten Projektion in die Instanzen eines XML-Schema Typs umgewandelt. Diese Projektion wird möglich durch die Verwendung von Meta-Modellen aus der M1-Schicht.

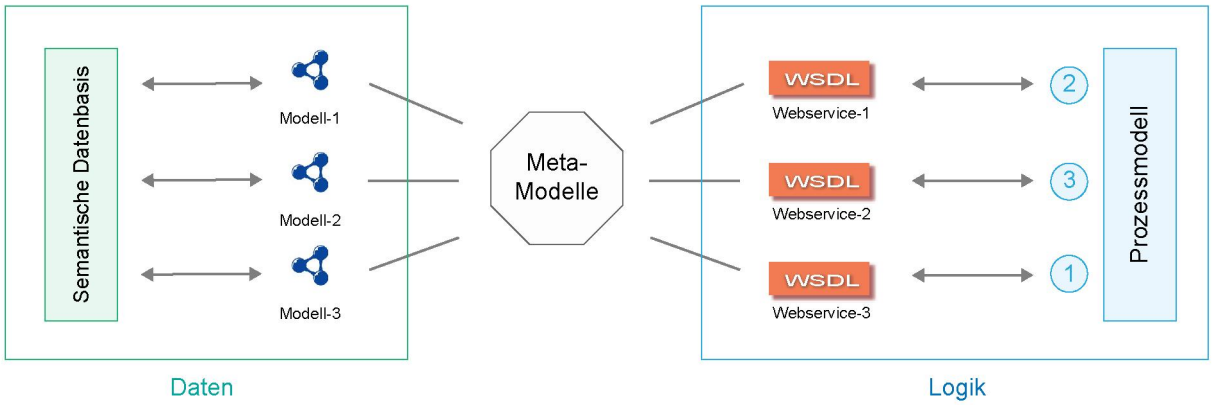

Abbildung 17 Konzept

In dieser Arbeit wurde diese Projektion mit Hilfe einer in Java festdefinierten Logik einprogrammiert. Jedoch werden aktuell einige interessante Frameworks entwickelt, die diese Aufgabe stark vereinfachen. Solche Frameworks funktionieren ähnlich wie die vergleichbaren Frameworks aus der klassischen Welt für die Abbildung von Informationen aus einer relationalen Datenbank in JavaBean Klassen. Ein gutes Beispiel dafür ist das Hibernate Framework (Hibernate, 2008). Derzeit ist JenaBean (Google) das bekannteste Framework für die Abbildung von Informationen eines RDF-Graphs in eine JavaBean Klasse. Dieses Framework wurde für einige einfache Projektionen in dieser Arbeit erfolgreich verwendet, jedoch war es für komplexe Abbildungen nicht geeignet. Deswegen wurde auf den Einsatz von JenaBean verzichten und für die Entkopplung der Abbildungslogik von dem Meta-Modell einer OWL-Klasse die SPARQL-Anweisungen verwendet. So enthalten die Webservice-1 bis -3 aus der Abbildung 17 die festprogrammierte Projektionslogik um die Modell-1 bis -3 aus der semantischen Datenbasis zu extrahieren und in eine Instanz eines XML-Schema Typs abzubilden.

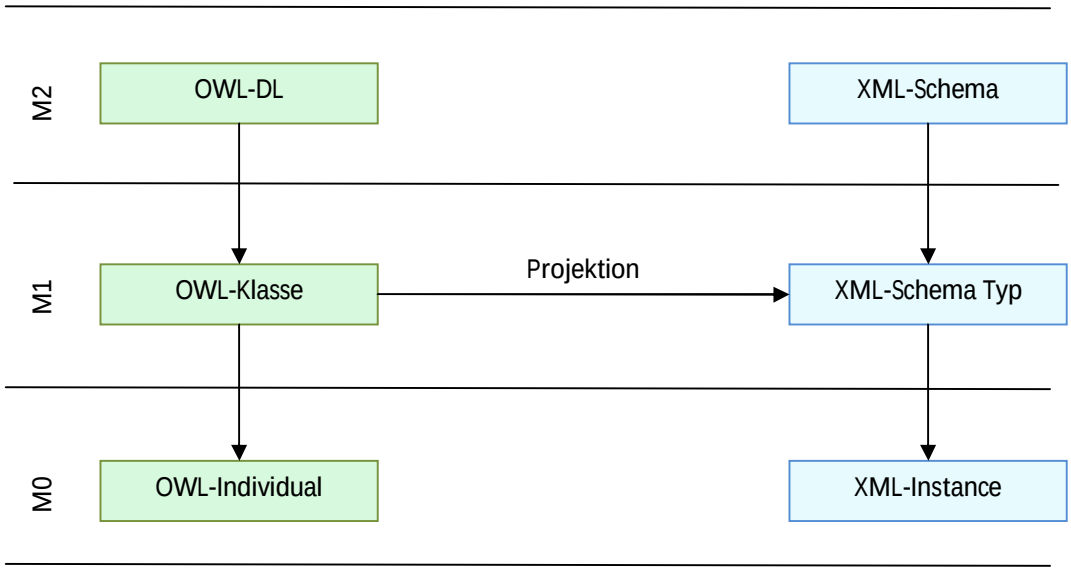

Abbildung 18 Abbildung von OWL-Individuals in XML-Schema Typen

2.3 Semantische Datenbasis

Jedes OWL-Dokument ist eine kleine semantische Datenbasis. Alle im Internet veröffentlichten OWL-Dokumente bilden zusammen eine gewaltige semantische Datenbasis. Das ist so, weil alle diese Dokumente theoretisch in einem einzigen RDF-Graphen eingelesen werden können. Dieser RDF-Graph würde alle in verschiedenen OWL-Dokumenten verteilten Informationen enthalten. Deswegen erweitert jedes in WWW veröffentlichte valide OWL-Dokument die gemeinsame globale semantische Datenbasis. Eine semantische Datenbasis unterscheidet sich von einer klassischen relationalen Datenbank in vielerlei Hinsicht:

1. ***Mehrschichten-Architektur***

   Das in OWL-Dokumenten enthaltende Wissen kann in Schichten aufgeteilt werden. Gewöhnlich werden die Schichten definiert um bestimmte Aspekte einer Anwendung abzukapseln. In dieser Arbeit wurden zwei Schichten definiert(Abbildung 19). Die erste Schicht (rot) kapselt den Aspekt der semantischen Klassifikation von Ressourcen ab. Sie bildet die Basisklassen für die ebenfalls während dieser Arbeit in OSPP implementierten facettenbasierten Klassifikation. Die zweite Schicht (grün) definiert spezielle Ressourcen, die von der in der ersten Schicht definierten Resource-Klasse erben und diese um zusätzlich benötigte Daten ergänzen. So wurde beispielsweise in der zweiten Schicht die OWL-Klasse für die Beschreibung der mathematischen Gleichung (Formula) definiert. Die Formula-Klasse erbt von der Resource-Klasse aus der ersten Schicht (rot) und erweitert diese um die Datenwerte, die für die Beschreibung einer mathematischen Gleichung nötig sind (siehe Abbildung 28 auf Seite 49). Es ist möglich, weitere Schichten zu definieren und die bestehende Datenbasis dadurch zu erweitern. So könnte beispielsweise die semantische Datenbasis in einer weiteren Schicht um die Zugriffsrechte der Nutzer auf verschiedene Ressourcen erweitert werden.

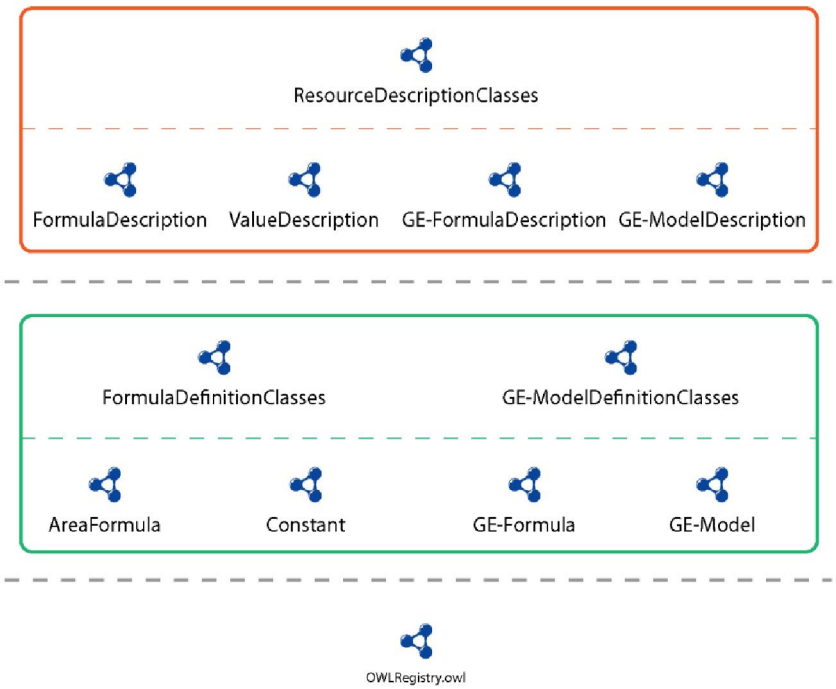

Abbildung 19 Schichten in einer semantischen Datenbasis

2. Modularisierte Datenbank

> *"Semantic technology helps computers understand data better. For businesses and other large organizations, this becomes particularly useful when merging large data sets. For example, merging two personnel databases when one defines only full-time employees and the other includes part-time or temporary workers can cause problems. Semantic technology can help resolve that."*
>
> *(Claburn, 2007)*

In einer relationalen Datenbank ist die Struktur der Datenbank durch das Entity-Relationship Modell starr festgelegt. Diese kann normalerweise an die neuen Anforderungen nur schwer angepasst werden. Ein Vergleich zwischen einem OWL-Dokument und einer relationalen Datenbank lässt eine Analogie erkennen: so entspricht das Entity-Relationship Modell einer relationalen Datenbank den OWL-Klassen, die binären Daten einer relationalen Datenbank den in OWL-Individuen (engl. Individuals) enthaltenen Informationen. Die OWL-Dokumente haben eine feste URL, deswegen können die Elemente eines OWL-Dokumentes über eine eindeutige URI im WWW referenziert werden. So können beispielsweise die OWL-Klassen eines OWL-Dokumentes in einem anderen OWL-Dokument durch das Referenzieren wiederverwendet werden. Jedes Individuum eines OWL-Dokumentes kann durch OWL in anderen Individuen referenziert werden und somit ein Teil deren Information bilden. Dadurch kann jedes Individuum oder ein anderes OWL-Element, das durch eine URI referenziert werden kann, als ein Modul gesehen werden, das in anderen Modulen (OWL-Elementen) verwendet werden kann. Abbildung 20 veranschaulicht die Beziehungen zwischen OWL-Elementen aus mehreren OWL-Dokumenten. Die Pfeile repräsentieren, welches Dokument in welchem importiert ist. So wird beispielsweise ***Dokument-1*** in dem ***Dokument-2*** importiert, wobei mehrere Elemente aus ***Dokument-1*** mehrere Elemente aus ***Dokument-2*** durch URI-Referenzen wiederverwenden können. ***Dokument-2 wird analog*** im ***Dokument-3*** und ***Dokument-3*** in dem ***Dokument-1*** importiert. Da ***Dokument-3*** in dem ***Dokument-1*** importiert ist, entsteht ein Zyklus zwischen mehreren OWL-Dokumenten. Im Semantic Web ist ein solcher Zyklus kein Sonderfall. Durch die ständig wachsende, selbstorganisierende Wissensbasis, werden viele OWL-Dokumente miteinander durch die Import-Anweisungen verknüpft. Die Auflösung solcher Zyklen ist im Regelfall die Aufgabe einer Inferenzmaschine. Das soll an einem einfachen Beispiel veranschaulicht werden: eine Inferenzmaschine lädt die ***Registry*** in einen RDF-Graph um eine SPARQL-Anfrage an diese zu stellen. Da die ***Registry*** das ***Dokument-2*** importiert, liest die Inferenzmaschine die Inhalte von ***Dokument-2*** ebenfalls in den RDF-Graphen ein. Das ***Dokument-2*** importiert jedoch ***Dokument-3***, dieses wiederum das ***Dokument-1***. ***Dokument-1*** importiert das ***Dokument-2***, ***Dokument-2*** wurde jedoch bereits in den RDF-Graphen geladen. Das erkennt auch die Inferenzmaschine und bricht die Ladeoperation ab, so werden die Zyklen zwischen mehreren RDF-Dokumenten unterbrochen. Eine Inferenzmaschine soll außer dem Erkennen und Abbrechen von Zyklen noch viele weitere Eigenschaften haben, zum Beispiel: Robustheit, effiziente Verarbeitung von besonders großen Datenmengen (Endres, et al., 2006). So kann eine Inferenzmaschine zum Beispiel die fehlenden Informationen aus RDF-Dokumenten, die durch das Ausfallen eines Servers im WWW nicht zugänglich sind, tolerieren. Jedes einzelnes OWL-Dokument bildet dabei einen Modul, der durch eine Import-Anweisung in einem anderen OWL-Dokument (Modul) wiederverwendet werden kann.

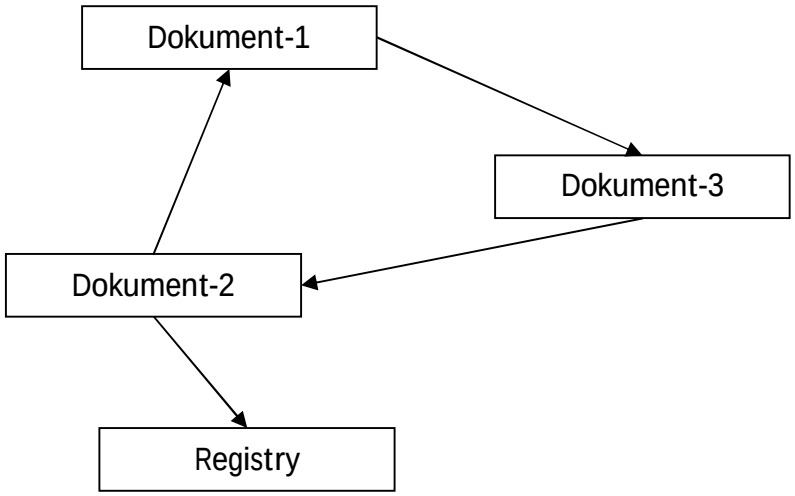

Abbildung 20 Erzeugung eines RDF-Graphen

3. Globale Datenbasis

Jedes im WWW veröffentlichte OWL-Dokument kann in anderen OWL-Dokumenten importiert und wiederverwendet werden. Verschiedene OWL-Dokumente können von verschiedenen Autoren erstellt worden sein. Da ein OWL-Dokument in mehreren anderen OWL-Dokumenten importiert sein kann, ändert ein Autor durch das Editieren seines OWL-Dokumentes das „Wissen" in allen OWL-Dokumenten, die dieses importieren. So könnte ein Bauingenieur ein Berechnungsmodell ändern, welches in mehreren anderen Berechnungsmodellen verwendet wird, und dadurch auch die anderen Berechnungsmodelle ändern. Das Konzept der Vererbung von OWL-Klassen ermöglicht eine hohe Wiederverwendung von Daten sowie eine enge Zusammenarbeit. Alle im WWW veröffentlichten OWL-Dokumente bilden eine Wissensbasis, die ständig durch das Verändern von bereits existierenden Dokumenten sowie durch das Hinzufügen von neuen verändert wird. Je länger ein RDF-Dokument veröffentlicht ist, desto mehr Verweise enthält die semantische Datenbasis auf dieses. Die Abbildung 21 veranschaulicht die durch die Selbstorganisation entstehende Komplexität einer semantischen Datenbasis. Viele Nutzer speichern ihr Wissen in der Datenbasis unabhängig von einander und verweisen dabei auf bereits existierendes Wissen eines OWL-Dokumentes eines anderen Nutzers. Die Komplexität der in WWW derzeit existierenden Datenbasis ist unvorstellbar, die Statistiken von Swoogle[16] sprechen für sich(Tabelle 1). Die während dieser Arbeit auf dem virtuellen Server veröffentlichten OWL-Dokumente, haben die unvorstellbar große WWW-Wissensbasis um einige Erkenntnisse aus dieser Arbeit erweitert. Insgesamt wurden während dieser Arbeit 42 OWL-Dokumente erstellt und in WWW veröffentlicht. Diese referenzieren nicht nur einander, sondern auch viele andere OWL-Dokumenten, wie beispielsweise die benötigten OWL-Dokumente für das Erstellen der OWL-S Beschreibung von Webservices.

[16] Swoogle ist eine Suchmaschine für semantische Dokumente http://swoogle.umbc.edu/

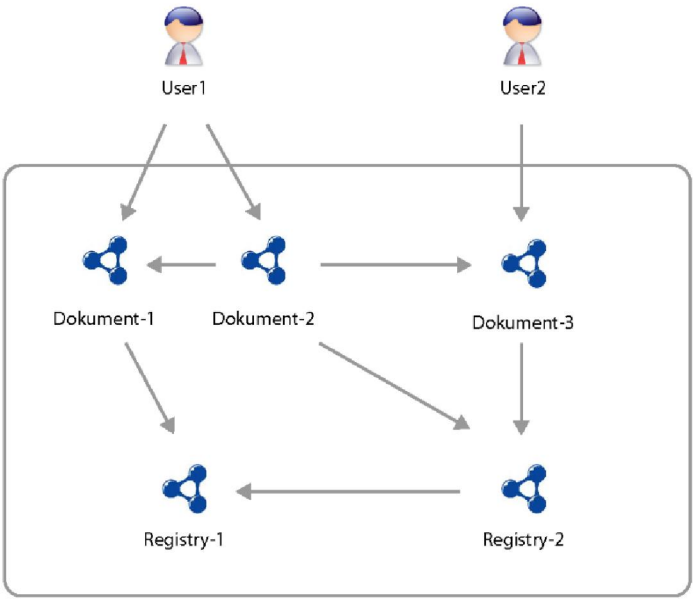

Abbildung 21 globale semantische Datenbasis

Datum Zeit	2008-10-13 23:58:51
Anzahl der entdeckten URLs	8.098.586
Anzahl der durch ping angesprochenen URLs	4.309.575
Anzahl der erreichten und bestätigten semantischen Dokumente	2.678.194
Anzahl von fehlerfreien semantischen Dokumente	1.500.912
Anzahl der Trippeln von allen geparsten semantischen Dokumenten	672.614.243

Tabelle 1 Statistik von Swoogle

2.4 Austauschen von Berechnungsmodellen

Im Abschnitt 2.1 wurde die Problematik der Bauwerksüberwachung verdeutlicht. Sie zeigte dass es notwendig ist die Modelle manchmal durch die andere Modelle zu ersetzen. Normalerweise werden Modelle in einer Software-Komponente implementiert und ermöglichen die Berechnung von speziellen Sachverhalten, so wird beispielsweise die Horizontalverschiebung einer Spundwand in dieser Arbeit von einem Webservice (Excavation-Service) berechnet. Ein solcher Webservice wird normalerweise an das Gesamtkonzept durch einen Adapter (Gamma, et al., 2001) angepasst, wobei der Adapter in einem Workflow Prozess mit zu einer Gesamtlogik orchestriert wird. Ein neuer Webservice mit einem anderen Berechnungsmodell kann durch eine neue Adapterimplementierung den verwendeten Webservice ersetzen. Das Austauschen so eines Webservices ist nicht trivial und erfordert Fachwissen über Workflow Prozesse. In der Abbildung 22 ist eine schematische Darstellung eines Stücks aus einem Workflow Prozess dargestellt. Dieser veranschaulicht die Verwendung eines Modells in einem Workflow Prozess über einen Adapter Service (***Adapter-WS***). Ein Adapter-WS sollte ein eigenständiger Webservice sein, der ein existierendes Berechnungsmodell durch die Implementierung einer in dem Workflow Prozess verwendeten Schnittstelle (WSDL) das Berechnungsmodell in dem Workflow Prozess austauschbar macht. Die Abbildung 23 stellt die Implementierung von Adapter-WS als

ein UML Diagramm dar. Das Berechnungsmodell könnte in mehreren Programmiersprachen von verschiedenen Entwicklern implementiert worden sein. Dabei können sich die Berechnungsmodelle voneinander stark unterscheiden und für verschiedene Situationen mehr oder weniger geeignet sein. In dem Abschnitt 2.1 vorgestellte Problematik beschreibt zum Beispiel die Verwendung eines Ingenieurmodells für die Berechnung der Horizontallverschiebung. Dieses Modell könnte beispielsweise in C#[17] von einem Entwickler implementiert (C#-Modell) und durch einen anderen Entwickler für die Verwendung in einem Workflow Prozess angepasst worden sein. Dafür hat der zweite Entwickler ein Adapter-Webservice (C#-WS) entwickelt, der die festgelegte Schnittstelle (Adapter-WS) implementiert(siehe Abbildung 23). Nach dem die vertikale Setzung eingetreten ist (siehe Abschnitt 2.1 Situation 2) muss das verwendete Berechnungsmodell (C#-Modell) ersetzt werden, da dieses die vertikale Setzung für die Vorhersage nicht berücksichtigen kann. Das andere Ingenieurmodell, das die vertikale Setzung berechnen kann, wurde von einem anderen Entwickler in Java implementiert (Java-Modell). Dieses Ingenieurmodell muss genauso wie C#-Modell durch einen Adapter Webservice (Java-WS) für eine Verwendung in einem Workflow Prozess zugänglich gemacht werden. Java-WS implementiert gleiche funktionale Schnittstelle (Adapter-WS) wie C#-WS. Durch die Verwendung einer gemeinsamen Schnittstelle (Adapter-WS), kann ein Ingenieurmodell (C#-WS) durch ein anderes (Java-WS) durch eine einfache Änderung des Workflow Prozesses (es müssen lediglich die Partner Links geändert werden) ersetzt werden. Das erfordert jedoch Fachwissen über die Modellierung eines Workflow Prozesses und ist mit hohem Deployment Aufwand verbunden, denn nach jeder Prozessänderung muss der Prozess neu deployed werden. Eine interessante Lösung für die Anpassung eines Ingenieurmodells bietet die Erstellung eines Workflow Prozesses, der die benötigte funktionale Schnittstelle (Adapter-WS) implementiert und andere im WWW existierenden Webservices, die das benötigte Ingenieurmodell anbieten, an die Adapter-WS Schnittstelle adaptiert. Der Vorteil dieser Vorgehensweise ist, dass ein Workflow Prozess in der Regel leichter als ein Webservice zu erstellen ist und bessere Möglichkeit für die Wartung anbietet. Während dieser Arbeit wurde prototypisch der Formula-Solver-Service (siehe Abschnitt 3.4) durch einen BPEL-Prozess adaptiert und in mehreren anderen BPEL-Prozessen als ein Unterprozess verwendet. Die Adapter-Webservices sollten nach dem Contract-First-Ansatz entwickelt werden und die gleichen XML-Schema Typen sowie die funktionale Schnittstelle haben. WSDL 2.0 ermöglicht zum Beispiel die Vererbung der funktionalen Schnittstelle und ist somit für diesen Ansatz besser als derzeit verbreiteter WSDL 1.1 Standard geeignet.

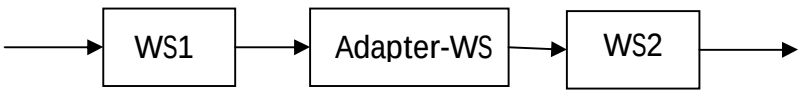

Abbildung 22 Adapter in einem Workflow Prozess

[17] C# ist eine Programmiersprache

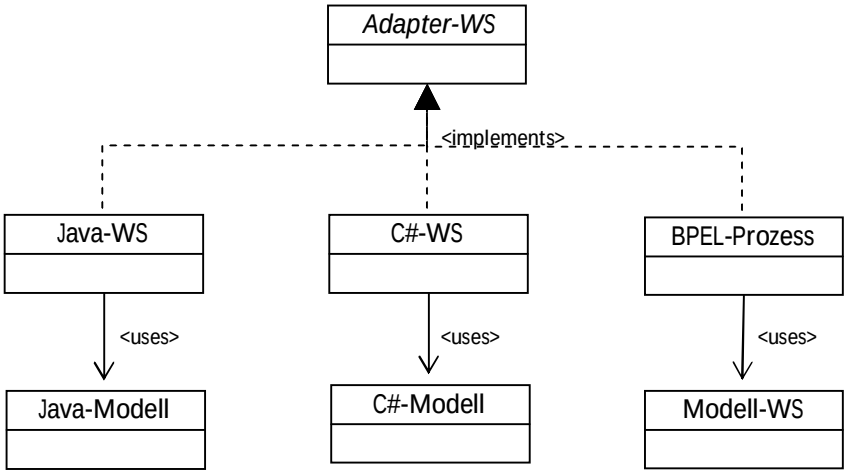

Abbildung 23 Implementierung von Adapter-WS

Ein anderer Ansatz für das Austauschen von Berechnungsmodellen ist viel generischer. Dabei wird das Berechnungsmodell in eine Datenstruktur ausgelagert. Diese Datenstruktur enthält die Berechnungslogik und kann von einer Software interpretiert werden. Diese Interpretationssoftware kann über einen Webservice für die Verwendung im WWW bereitgestellt werden. So kann eine Datenstruktur, die ein Ingenieurmodell repräsentiert, zu einem Webservice hochgeladen und dort ausgeführt werden. Dieser Ansatz erinnert an das Konzept von Apache Axis(Apa08). Bei Axis wird die Logik eines Webservices in einen Axis Archive(.aar) ausgelagert. Die Axis Archive können über die Axis-Webapplikation zum Server hochgeladen und dort veröffentlicht werden (Deployment). Nach dem Deployment kann in Axis Archiven enthaltene Webservice-Logik mit Hilfe eines dafür von Axis automatisch erstellten Webservice ausgeführt werden. Axis kapselt dabei den Aspekt von SOAP-Nachrichten Kodierung und Dekodierung von der Business Logik ab. Interessant ist auch, dass die Axis Laufzeitumgebung selbst als eine Webapplikation in einem Servlet Container wie Tomcat laufen kann. Ein Webapplikation Archiv (.war) wird auf einem J2EE Servlet Container deployed. Der Servlet Container implementiert dabei den Aspekt einer J2EE Applikation und ermöglicht zum Beispiel die Stellung von HTTP-Anfragen an eine Webapplikation. Ein Webservice, der das Hochladen eines Berechnungsmodells erlauben soll, muss den Berechnungsaspekt eines Ingenieurmodells von der internen Logik eines Webservices abkapseln. In der Abbildung 24 ist das Berechnungsmodell mit Hilfe von OWL-DL beschrieben und in einem OWL-Dokument gespeichert. Dieses Modell kann auf dem speziell dafür programmierten Webservice (Modell-WS) berechnet werden. Dieser Webservice kapselt die Business Logik für das Laden und Parsen eines in einem OWL-Dokument gespeicherten Ingenieurmodells von deren Berechnungslogik ab. Der Webservice (Modell-WS) wiederum läuft auf der Axis2-Engine. Und Axis2 läuft als eine Webapplikation in einem Tomcat 5.5 Servlet Container. Es gibt mehrere Vorteile, die bei der Auslagerung von Modellen in einen OWL-Dokument, entstehen. Zum Einen kann ein Nutzer das Modell recht einfach ersetzen, ohne dafür etwas programmieren zu müssen. Zum Anderen ist OWL einer der Standards des Semantic Webs, deswegen können die Modelle, die in OWL beschrieben sind, schnell gefunden werden. Die Suche von Modellen kann über eine Benutzeroberfläche oder über eine Abfragesprache wie SPARQL erfolgen.

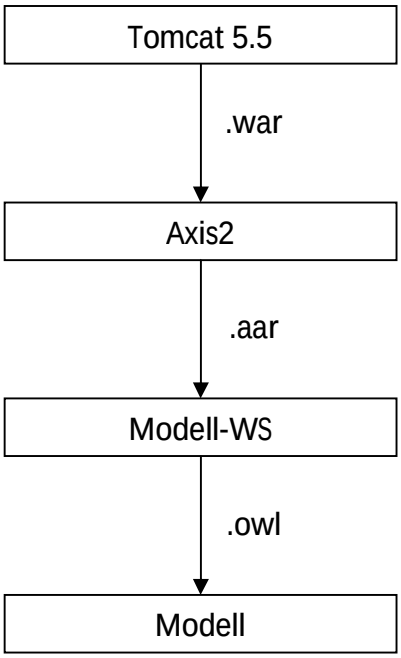

Abbildung 24 Auslagern von Logik

2.5 Facettenbasierte Klassifikation

In der Abbildung 25 ist das während dieser Arbeit entwickelte Konzept für die semantische Suche von Webservices mit Hilfe von Facetten veranschaulicht. In der unteren Schicht (Impl) befindet sich die Implementierung eines Webservices, zum Beispiel ein in Java programmierter Axis2 Webservice. Dieser Webservice ist in der OWL-S Schicht mit Hilfe des OWL-S Standard semantisch beschrieben. Es können mehrere OWL-S Beschreibungen eines Webservices existieren, die einen Webservice aus verschiedenen Sichten semantisch beschreiben. So beschreibt ein Datenbankexperte einen Webservice aus der Sicht der Sicherheit und Transaktionen, wobei ein Wirtschaftsinformatiker den gleichen Webservice aus der Sicht des wirtschaftlichen Nutzens beschreibt. Ein Webservice kann mit Hilfe von Kategorien im Profile des OWL-S Standards klassifiziert werden. Die Kategorien können durch ihre URIs im WWW referenziert und so in anderen OWL-S Beschreibungen wiederverwendet werden. So können beispielsweise einige Kategorien des Service-A aus der Abbildung 25 in der Klassifikation des Service-B durch das Referenzieren wiederverwendet werden. Da OWL-S Beschreibung eines Webservices ein gewöhnliches OWL-Dokument ist, können die für die Klassifikation benötigten Kategorien auch in einem eigenen OWL-Dokument definiert werden und später in mehreren OWL-S Beschreibungen(Service-A, Service-B) für die Klassifikation von Services verwendet werden. Durch die Klassifikation von mehreren Dingen mit gleichen Kategorien (eine Kategorie wird durch eine URI eindeutig beschreiben) entsteht eine Art gemeinsame Sprache, die es möglich macht die Suchontologie für facettenbasierte Suche in der oberen Schicht (Facetten) festzulegen. Die Suchontologie definiert die Facetten, in die die Kategorien einsortiert werden. Das sind die gleichen Kategorien, die für die Klassifikation von Webservices verwendet wurden und in dem OWL-S Profile von der semantischen Beschreibung eines Webservices enthalten sind. Normalerweise sollten die Facetten eine flache Ordnerstruktur darstellen, wobei in jedem Ordner viele Kategorien enthalten sein können, jedoch nur eine ausgewählt werden darf. Die

Entwicklung solcher Facetten ist oft sehr umständlich und stößt in der Praxis schnell an ihre Grenzen.

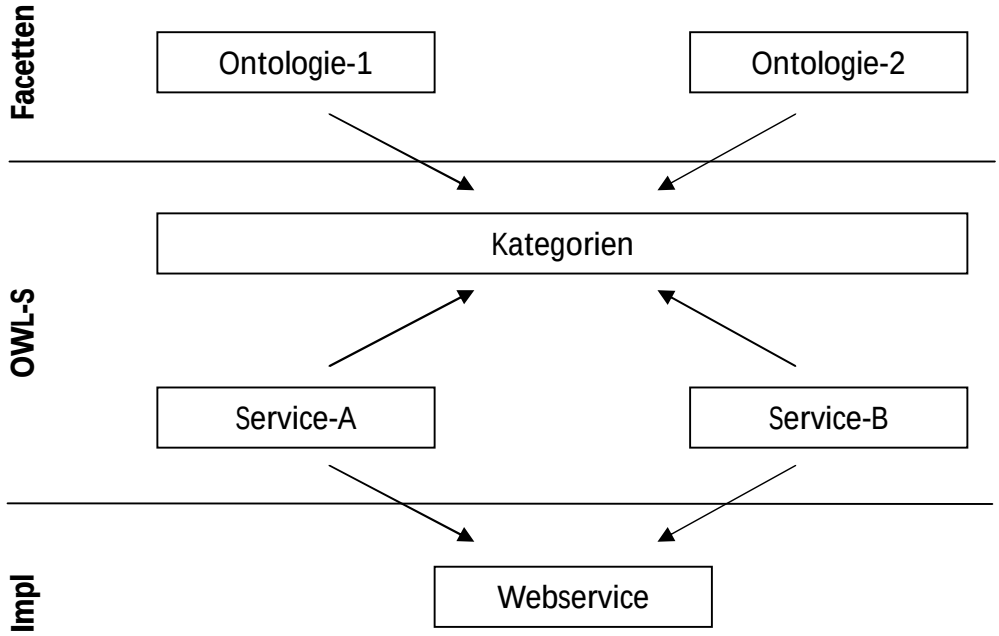

Abbildung 25 Facetten-Basierte Klassifikation eines Webservices

In (Mias, 2000) wird diskutiert ob das Wissen sinnvoll geordnet werden kann, denn die Wissensordnung hängt stark mit der Weltanschauung zusammen. Jeder Mensch nimmt seine Umwelt anders wahr als ein anderer. Die Wahrnehmung ist subjektiv und somit ist auch der Erwerb des Wissens (die Suche nach dem Wissen ist subjektiv). Der Inhalt einer Klassifikation bildet auch einen Teil des Wissens und somit ist die Klassifikation auch subjektiv. Deshalb sollte die Wissensordnung (definierte Facetten) möglichst unabhängig von den subjektiven Vorverständnissen eines Nutzers sein (Mias, 2000). Aus diesem Grund wurde in dieser Diplomarbeit für die Suche eine Suchontologie mit unbegrenzter Tiefe und mit mehrfacher Auswahl pro Facette implementiert. Das heißt, eine Facette kann nicht nur Kategorien, sondern auch weitere Facetten (Unterfacetten) enthalten. Es können mehrere Kategorien in einer Facette und in deren Unterfacetten ausgewählt werden. Damit verschiedene Nutzer verschiedene an ihre Bedürfnisse angepasste Suchen durchführen können, erlaubt das Programm das Laden von verschiedenen Suchontologien zur Laufzeit. So kann beispielsweise ein Bauingenieur der an der Aufgabe-1 arbeitet die Suchontologie-1 für die Suche von Webservices verwenden und für die Aufgabe-2 die Suchontologie-2. Die Suchontologie-1 und -2 können verschiedene Facetten enthalten, die für die Suche von verschiedenen Webservices geeignet sind, außerdem können die gleichen Facetten unterschiedlich sortiert werden und damit die Entscheidungsbäume nachbilden und die Suche nach benötigten Ressourcen optimieren. In der Abbildung 43 (siehe Anhang) ist beispielsweise ein Suchdialog mit einer Suchontologie abgebildet, der einen Entscheidungsbaum für die Suche einer mathematischen Formel nachbildet. Zuerst wird nach dem maßgeblichen Erddruckzustand gefragt (ruhe, passiv, aktiv), dann nach der Bodenart (nichtbindig, bindig), dann nach Ursache und so weiter, bis der

Nutzer, die benötigte mathematische Formel findet. Das Laden eines Suchbaumes zur Laufzeit des Programms löst das Problem der verschiedenen Weltanschauung von verschiedenen Nutzern. Zum Beispiel gibt es in einer Bibliothek zwei Arten von Nutzern: (1) der aktive Nutzer (Bibliothekar) und (2) der passive Nutzer (Besucher). Die beiden Nutzer sprechen miteinander in einer künstlichen Sprache, mittelbar durch eine Klassifikation über den Inhalt der Bücher (Nohr, 1996). Die beiden Nutzer haben unterschiedliches Vorwissen, jedoch müssen Sie sich an einem in der Bibliothek für alle vordefinierten Suchbaum orientieren. Dadurch gestaltet sich die Suche nicht immer optimal, da diese nicht an dem Vorwissen des Suchenden angepasst ist. In der virtuellen Welt ist es möglich die Suchbäume dynamisch zu laden und somit für eine bestimmte Zielgruppe anzupassen. So können nicht nur ein Bibliothekar und ein Besucher verschiedene Sichten auf die Inhalte einer Bibliothek haben, sondern auch verschiedene Besucher. Zum Beispiel ein Kind, das ein Kinderbuch über Pooh den Bären braucht und ein Physiker, der ein Buch über die Quantenmechanik sucht.

3. Implementierung

Dieser Kapitel behandelt im ersten Abschnitt die Implementierung des in Abschnitt 2.2 vorgestellten Konzeptes. Danach werden die einzelnen Komponenten, sowie die dazu nötigen Technologien beschrieben, wie semantische Datenbasis, Adapter- und Funktionale-Webservices. Im Anschluss wird das Evaluierungsbeispiel vorgestellt, das die im Abschnitt 2.1 vorgestellte Problematik aus dem Bauwesen löst und alle zuvor in diesem Kapitel beschriebenen Komponenten verwendet. Nach dem Evaluierungsbeispiel werden die während dieser Arbeit entwickelten OSPP-Erweiterungen vorgestellt. Dieses Kapitel schließt mit einer Diskussion über die Implementierung einer facettenbasierten Suche sowie über die Reife der verwendeten Werkzeuge und ihren möglichen Einsatz in der Praxis.

3.1 Implementierung des Konzeptes

Das Konzept aus der Abbildung 17 (siehe Abschnitt 2.2) wurde in dieser Arbeit unter der Verwendung aktueller Technologien prototypisch umgesetzt. Die semantische Datenbasis wird dabei durch mehrere im Internet veröffentlichten OWL-Dokumente realisiert (Abbildung 26). Die Abbildung von Informationen aus OWL-Dokumenten geschieht mit Hilfe speziell dafür entwickelter Webservices, den sogenannten Adapter Webservices. Diese extrahieren die Informationen aus OWL-Dokumenten und bilden diese in die XML-Schema Typen ab. Alle während dieser Arbeit entwickelten Adapter-Webservices ermöglichen außerdem eine semantische Suche von in XML-Schema Typen abzubildenden Ressourcen. So gibt es beispielsweise einen Webservice, der eine mathematische Formel in der semantischen Datenbasis suchen und diese als XML-Schema Typ in einem Workflow-Prozess bereitstellen kann. Die Adapter Webservices können auf verschiedene Weise genutzt werden, zum Beispiel für die Suche von benötigten Formeln in einer Entwicklungsumgebung (zum Beispiel OSPP), aber auch für die Berechnung einer mathematischen Gleichung in einem Workflow Prozess. Die im Konzept dargestellten funktionalen Webservices enthalten die für die Berechnung von Bauprozessen benötigte Logik. So kann beispielsweise ein während dieser Arbeit implementierter Webservice (Excavation-Service) die Horizontalverschiebung einer Spundwand berechnen (siehe Abschnitt 2.1). Die erstellten Adapter- und Funktionale Webservices werden in Workflow Prozessen zu einer komplexen Geschäftslogik orchestriert. Ein gutes Beispiel dafür bietet der Evaluierungsbeispiel aus dem Abschnitt 3.5.

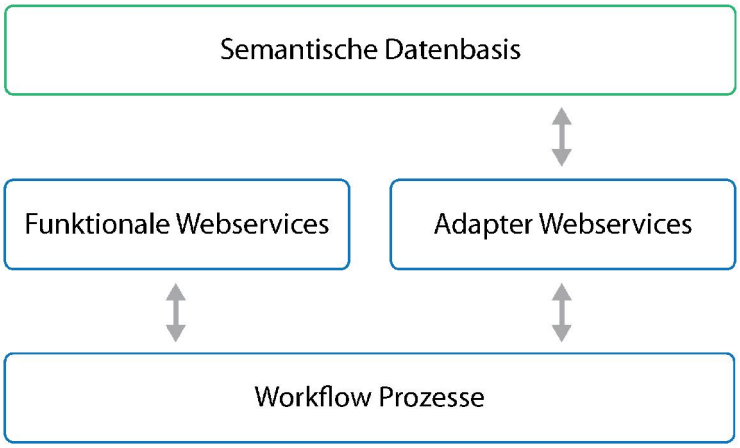

Abbildung 26 Konzept Implementierung

3.2 Semantische Datenbasis

Die während dieser Arbeit erstellten OWL-Dokumente bilden gemeinsam die in dieser Arbeit verwendete semantische Datenbasis. Die semantische Datenbasis enthält verschiedenste Informationen wie Formeln, statische Baumodelle, semantische Beschreibungen von Webservices, die Suchontologien für die facettenbasierte Suche in OSPP und vieles mehr. Alle OWL-Dokumente werden in der ***OWLRegistry.owl*** importiert (Abbildung 27). OWLRegistry.owl ist ein OWL-Dokument das durch die Imports aller während dieser Arbeit erstellten OWL-Dokumente, das gesamte Wissen aus der semantischen Datenbasis bündelt. Somit können alle Suchanfragen an dieses eine OWL-Dokument gestellt und die benötigten Daten aus der semantischen Datenbasis rausgeholt werden. In der Abbildung 27 ist die prototypische

Umsetzung der semantischen Datenbasis veranschaulicht. Alle OWL-Dokumente können im WWW verteilt auf verschiedenen Servern vorhanden sein. Die Dokumente können durch URIs die Beziehungen zueinander definieren und somit das bereits existierende Wissen aus referenzierten OWL-Dokumenten nutzen, um eigenes zu beschreiben. So werden beispielsweise die Attribute aus *Formula-Description.owl* in dem *AreaFormula.owl* genutzt, wobei die in *Formula-Description.owl* definierten Attribute für die Klassifikation von Formeln in *AreaFormula.owl* genutzt wurden. Die in der Abbildung 27 veranschaulichte semantische Datenbasis besitzt zwei Schichten. Die obere Schicht (rot) ermöglicht die Facettenklassifikation von Ressourcen, diese macht es möglich die klassifizierten Ressourcen mit Hilfe von facettenbasierten Suche in OSPP semantisch zu suchen.

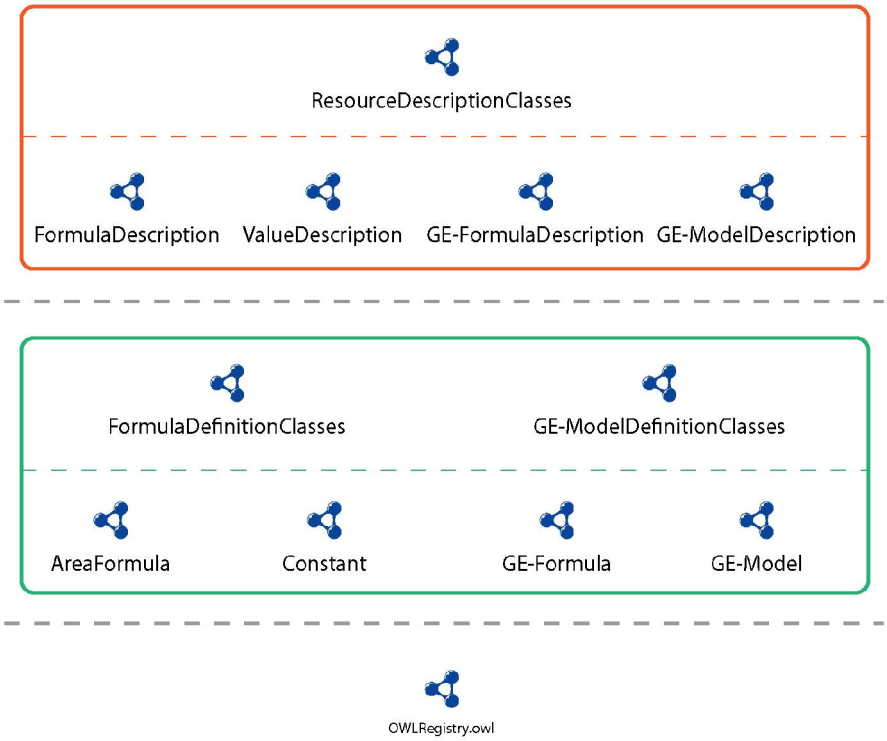

Abbildung 27 Semantische Datenbank enthält mehrere Schichten und ist von zwei Personen erstellt worden.

Die untere Schicht (grün) enthält spezielle Ressourcen, wie beispielsweise mathematische Formeln oder statische Datenmodelle. Die gestrichelte Linie trennt jede Schicht in die Klassendefinition und OWL-Individuen. Zum Beispiel definiert die *ResourceDescriptionClasses.owl* die nötigen OWL-Klassen für Facettenklassifikation einer Ressource, wobei die *FormulaDescription.owl* einige konkrete Individuen der Attribute-Klasse enthält. Diese Attribute werden später für die Facettenklassifikation der Formeln in mehreren OWL-Dokumenten verwendet, zum Beispiel in *AreaFormula.owl* oder *GE-Formula.owl*. Die oberen OWL-Dokumente aus der unteren Schicht enthalten die Klassendefinitionen von bestimmten Ressourcen. Zum Beispiel enthält *FormulaDefinitionClasses.owl* die Klassendefinition einer Formel. Die Individuen von dieser Klasse werden in *AreaFormel.owl* und *GE-Formula.owl* angelegt. Die OWL-Dokumente einer semantischen Datenbasis werden von mehreren Nutzern unabhängig voneinander angelegt, deswegen hat die semantische Datenbasis genauso wie das WWW den selbstorganisierenden Charakter (siehe Abschnitt 2.3). Nun sollen die Beziehungen von zwei oben beschriebenen Schichten anhand eines RDF-Graphen genauer erläutert werden. In der Abbildung 28 sind einige Relationen zwischen einigen wichtigen Klassen aus der semantischen Datenbasis veranschaulicht. Die

Resource-Klasse steht in Beziehung *hasAttribute*-Relation zur *Attribute*-Klasse und kann keinen oder mehrere Attribute haben. Die *Resource* und *Attribute* Klassen bilden das „Fundament“ der ersten Schicht, denn alle benutzerspezifischen Ressourcen aus der zweiten Schicht stehen in *isa*-Relation zu der *Resource*-Klasse aus der ersten Schicht und erben ihre Funktionalität. So erweitert die *Formula*-Klasse die *Resource*-Klasse um die benötigten Daten, wie Eingabe- oder Ausgabe-Parameter. Eingabe und Ausgabe Parameter sind ebenfalls von der *Resource*-Klasse abgeleitet und können ebenfalls in der semantischen Suche einer Formel berücksichtig werden. So kann beispielsweise eine Formel mit einem positiven Rückgabewert gesucht werden.

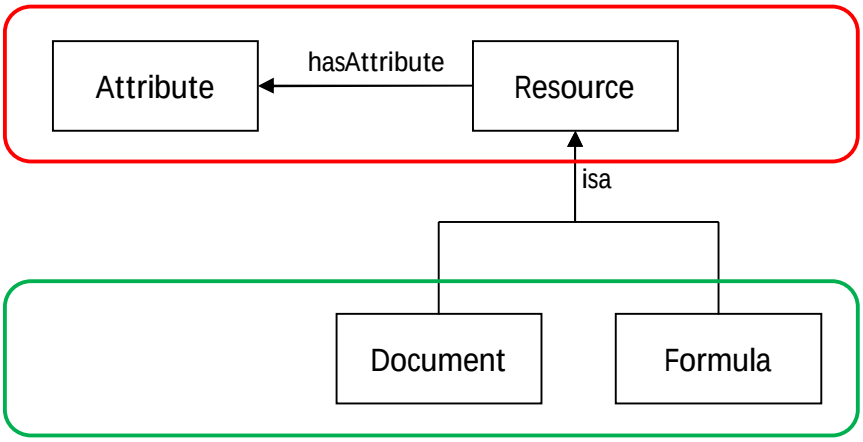

Abbildung 28 OWL-Klassen und ihre Beziehungen in dem RDF-Graph

3.3 Adapter Webservices

Die semantische Datenbasis kann viele verschiedene Informationen enthalten. Damit diese in einem BPEL-Prozess verwendet werden können, müssen sie in einen XML-Schema Typ abgebildet werden. Diese Abbildung geschieht in einem sogenannten Adapter-Webservice. In dieser Arbeit wurden mehrere Adapter-Webservices für verschiedene XML-Schema Typen implementiert. Die Architektur eines Adapter-Webservices soll anhand des *Formula-OWL-Adapter-Service* in der Abbildung 29 verdeutlicht werden. Der *Formula-OWL-Adapter-Service* kann die Formeln aus der semantischen Datenbasis extrahieren und diese als XML-Schema Typen für eine Verwendung in einem Workflow Prozess bereitstellen. *Formula-OWL-Adapter-Service* verwendet die Jena-API (Jen08) für das Parsen von OWL-Dokumente aus der semantischen Datenbasis und nutzt außerdem SPARQL für die Suche von benötigten Informationen. Alternativ zu SPARQL könnte die Abbildungslogik auch in Java unter der Nutzung der Jena-API festeinprogrammiert werden. Die Verwendung von SPARQL ermöglicht jedoch eine bessere Entkopplung zwischen Java und OWL. Dadurch sind eine schnellere Anpassung der Abbildungslogik nach der Änderung von OWL-Klassen in der semantischen Datenbasis sowie eine bessere Wiederverwendung des entwickelten Codes möglich. Alternativ zur Jena-API könnte auch die OWL-API (The081) genutzt werden, jedoch sollte dabei berücksichtigt werden, dass die OWL-API aktuell noch nicht die Ausführung von SPARQL Anfragen unterstützt. Manchmal ist es nötig, nicht einen bestimmten Typ aus der semantischen Datenbasis zu extrahieren, sondern es werden bestimmte Informationen aus der semantischen Datenbasis benötigt. Zum Beispiel die Höhe der Spundwand für die Berechnung der

Horizontalverschiebung. Dabei muss der Nutzer in der Lage sein, die benötigten Informationen mit Hilfe einer Abfragesprache (SPARQL) aus der semantischen Datenbasis zu extrahieren und diese in dem Workflow Prozess für die Berechnung bereitzustellen. Um das zu ermöglichen, wurde während dieser Arbeit ein Adapter-Webservice entwickelt, der das Stellen SPARQL-Anfragen ermöglicht und die Ergebnisse in XML-Schema Typ abbildet, sodass bei Bedarf eine SPARQL-Anfrage aus einem Workflow Prozess heraus gestellt werden kann und deren Ergebnisse weiter verarbeitet werden können. In der Abbildung 30 ist eine einfache SPARQL-Anfrage abgebildet, die eine Ressource aus einem RDF-Dokument extrahiert.

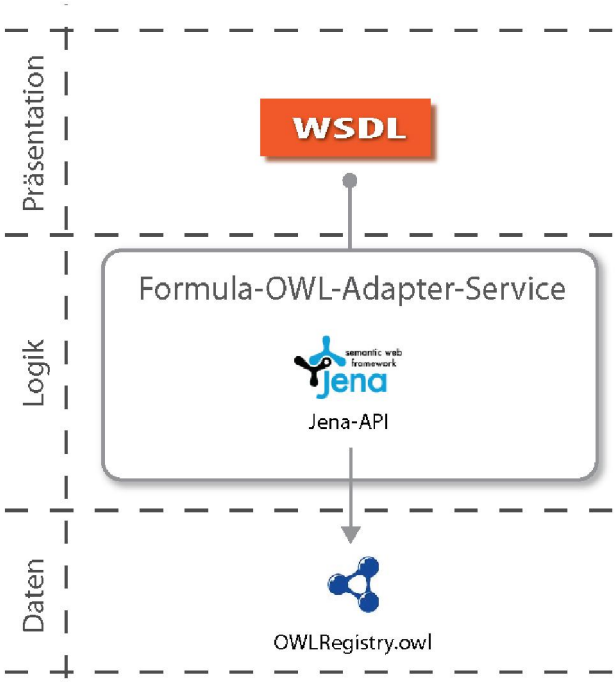

Abbildung 29 Formula-OWL-Adapter-Service

```
PREFIX rdc: <http://141.30.165.8:9762/webdav/ResourceDescriptionClasses.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ddc: <http://141.30.165.8:9762/webdav/Documents/description/DocumentDefinitionClasses.owl#>
SELECT ?ressource ?docURL ?docName
WHERE
{
?ressource rdf:type ddc:Document.
FILTER(?ressource=<http://141.30.165.8:9762/webdav/Documents/description/DocumentDescription.owl#Ablaufsplan.ifc>).
?ressource ddc:hasDocumentURL ?docURL.
?ressource rdc:hasName ?docName.
}
```

Abbildung 30 Die SPARQL Anfrage extrahiert eine Ressource aus einem RDF-Dokument

3.4 Funktionale Webservices

Für komplexe Berechnungen im Bauwesen sind oft viele Einzelberechnungen notwendig. Die Formeln für diese Berechnungen können zum Beispiel direkt in Java implementiert und als Webservice veröffentlicht werden. Solche Webservices gehören zu funktionalen Webservices und können in einem komplexen Workflow Prozess zusammen zu einer komplexen Geschäftsprozesslogik orchestriert werden. Ein Beispiel für einen solchen Webservice bietet der während dieser Arbeit entwickelte *Excavation-Webservice*, der die Logik für die Berechnung der Horizontalverschiebung aus in dem Abschnitt 2.1 beschriebenen Problemantik implementiert. Im Bauwesen gibt es viele Sachverhalte, die berechnet werden müssen, was die Entwicklung von sehr vielen Webservices nötig machen würde. Wobei jeder Webservice eine bestimmte Problemstellung lösen kann. Das führt zu einer sehr hohen Webserviceanzahl, die schwer zu verwalten ist und schlecht wiederverwendbar ist. Außerdem enthalten die entwickelten Workflow Prozesse viele Web Referenzen auf viele Webservices und werden dadurch schwer begreifbar. Deswegen wurde in dieser Arbeit ein Ansatz gewählt, einen generischen Webservice für die Berechnungsmodelle zu erstellen und die mathematische Berechnung als Aspekt aus dem Webservice auszulagern (siehe Abschnitt 2.4). Der generische Ansatz hat sich während dieser Arbeit gelohnt, es führte zu kürzerer Entwicklungszeit, besserer Wartung, sowie höherer Software-Qualität. Ein Beispiel eines generischen Webservices stellt der *Formula-Solver-Service* dar. Dieser ermöglicht die Berechnung von mathematischen Gleichungen mit Hilfe eines Mathematik Parsers (Expression4J (Exp081)). Expression4J ist ein Open Source Mathematik Parser. Er implementiert viele grundlegende mathematische Funktionen wie Sinus, Cosinus oder Logarithmus. Expression4J hat eine offene API und kann um benutzerspezifische Funktionen erweitert werden. Für Differenzial- und Integralrechnung kann GattMath(Gat) verwendet werden. GathMath ist ähnlich wie Expression4J ein Open Source Projekt. Es ermöglicht die Berechnung von Integralen und Differenzialen. Bei Bedarf von Diferential- und Integralrechnung sollte eine Einbindung von GattMath als benutzerspezifische Funktion in Expression4J untersucht werden. Eventuell kann die bereits in *Formula-Solver-Service* angepasste Funktionalität von Expression4J um Integral- und Diferenzialrechnung ergänzt werden. Aus Performance Gründen werden die zu berechnenden Formeln auf dem *Formula-Solver-Service* in einer relationalen Datenbank zwischengespeichert. Die verwendete relationale Datenbank wird über das Hibernate Framework verwaltet. Dadurch wird die Datenhaltung von der Geschäftslogik des Webservices entkoppelt. So kann beispielsweise die in dieser Arbeit verwendete HSQL-Datenbank gegen eine andere nur durch das Verändern der Konfigurationsdatei von dem Hibernate-Framework ausgetauscht werden. Der *Proxy-Formula-Solver-Service* evaluiert das Konzept des Austausches eines Berechnungswebservices gegen einen anderen aus dem Abschnitt 2.4. *Proxy-Formula-Solver-Service* ist ein BPEL-Prozess, der eine bestimmte WSDL-Schnittstelle implementiert. Wenn nun anstelle des Formula-Solver-Service ein anderer Webservice für die Berechnung der mathematischen Formeln verwendet werden soll, kann ein neuer BPEL-Prozess entwickelt werden, der die gleiche WSDL-Schnittstelle implementiert und die Operationsaufrufe an einen anderen Webservice weiterleitet. Ein vereinfachter Kontrollfluss einer Operation des *Proxy-Formula-Solver-Service* ist in der Abbildung 32 dargestellt. Die abgebildete Operation soll eine einfache mathematische Gleichung $f(a,b)=a+b$ mit Parametern $a=3$ und $b=2$ berechnen. Die zu berechnende Gleichung sowie die Parameter werden der *execute-formula()* Operation des *Proxy-Formula-Solver-Service* übergeben. Diese Operation ist als ein BPEL-Unterprozess implementiert, sie ruft mehrmals verschiedene Operationen des ***Formula-Solver-Service auf.*** Zuerst wird die Methode *set-formula()* aufgerufen. Diese Methode speichert die übergebene

Gleichung *f(a,b)=a+b* unter einer festgelegten *id* in einer relationalen Datenbank des *Formula-Solver-Services*. Danach lässt *Proxy-Formula-Solver-Service* die zuvor gesetzte Formel mit übergebenen Parametern *a=3, b=2* mit Hilfe von der Operation *execute-formula()* berechnen.

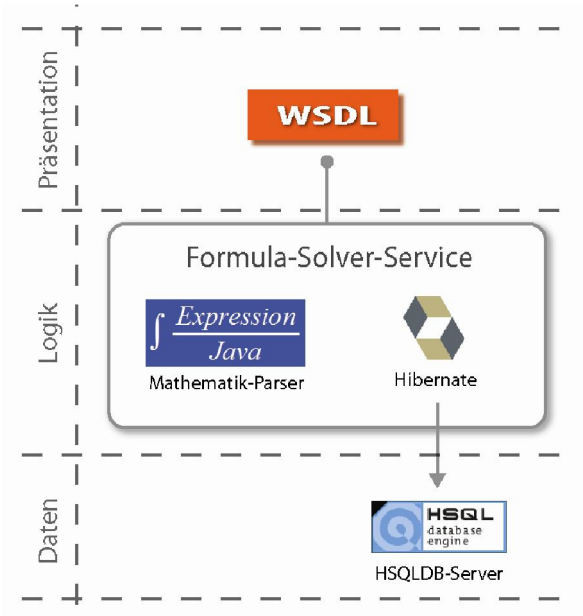

Abbildung 31 Formula-Solver-Service

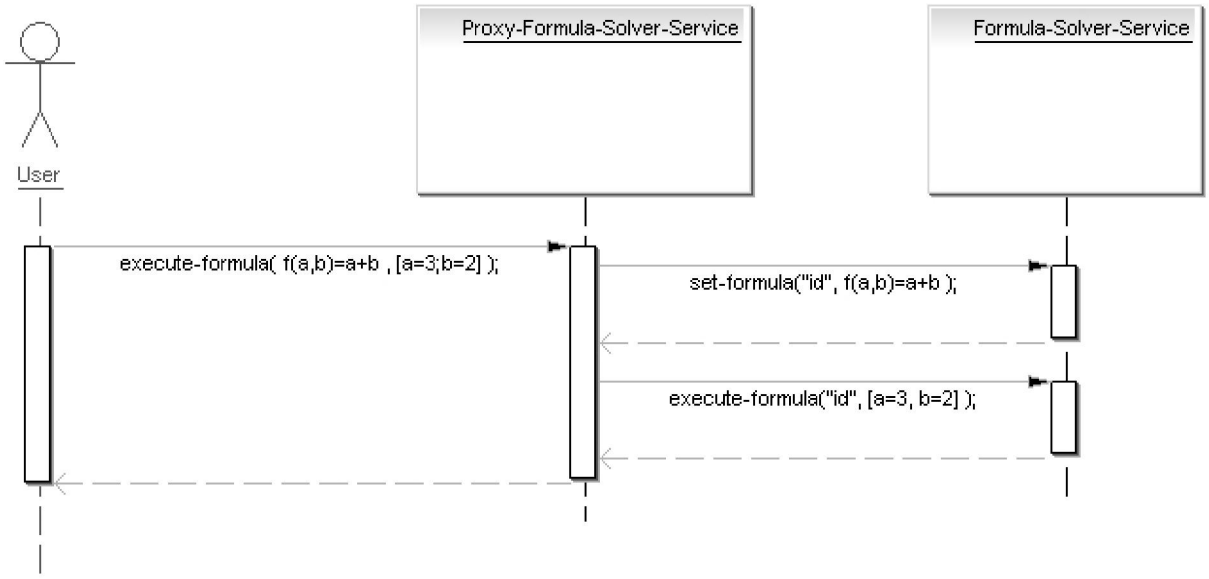

Abbildung 32 Proxy-Formula-Solver-Service

3.5 Evaluierungsbeispiel

In diesem Abschnitt werden die Vorteile des im Abschnitt 2.2 beschriebenen Konzepts anhand eines Evaluierungsbeispiels anschaulich gemacht. Die Abbildung 33 veranschaulicht die während dieser Arbeit implementierten und in dem Evaluierungsbeispiel verwendeten Komponenten. Die obere Schicht gehört zur semantischen Datenbasis. Sie enthält die Formeln und die Datenmodelle von Bauwerken (zum Beispiel von einer Baugrube). Die obere Schicht wird analog zu einer Datenschicht in einer klassischen Drei-Tier-Architektur im Evaluierungsbeispiel verwendet. Die mittlere Schicht enthält die funktionalen Webservices und Adapter-Webservices. Zum Beispiel *Formula-Solver-Service* ist ein funktionaler Webservice und *Formula-OWL-Adapter-Service* ein Adapter Webservice. Die Untere Schicht enthält Workflow Prozesse, die die Webservices aus der mittleren Schicht orchestrieren und ein bestimmtes Problem aus dem Bauwesen lösen. So berechnet beispielsweise der BPEL-Prozess aus diesem Evaluierungsbeispiel die Horizontalverschiebung einer Spundwand und löst somit die in dem Abschnitt 2.1 beschriebene Problematik. Das Evaluierungsbeispiel demonstriert zwei möglichen Arten einer Verwendungen eines Berechnungsmodells (siehe Abschnitt 2.4). Zum Einen den generischen Ansatz wo die Berechnungsmodelle (Formeln) aus der semantischen Datenbasis geladen werden und dann auf einem Mathematik Webservice berechnet werden können und zum Anderen einen Adapter-Ansatz wo ein Webservice (*Excavation-Service*) mit Hilfe eines BPEL-Prozesses an eine WSDL-Schnittstelle angepasst werden kann und während des Deployment Prozesses gegen einen anderen ausgetauscht werden kann. So wurde ein Teil der Berechnungslogik mit Hilfe von Formeln aus der semantischen Datenbasis realisiert und ein anderer mit Hilfe eines Webservices (*Excavation-Service*). Die Formeln wurden mit Hilfe des *Formula-OWL-Adapter-Services* aus der semantischen Datenbasis extrahiert und in dem BPEL-Prozess als komplexe XML-Schema Datentypen bereitgestellt. Der BPEL-Prozess konnte diese XML-Daten mit XPath transformieren und einen SOAP-Request an den Formula-Solver-Service stellen. Der Formula-Solver-Service hat die Formeln berechnet und diese Ergebnisse wurden weiter in der Logik des BPEL-Prozesses für weitere Berechnungen verwendet. Da einige Formeln in einem BPEL-Prozess manchmal mehrmals mit verschiedenen Parametern berechnet werden müssen, werden diese von dem *Formula-Solver-Service* in einer relationalen Datenbank zwischengespeichert und müssen nicht immer wieder in einem SOAP-Request übertragen werden. Die Formeln werden über *Formula-Solver-Service* unter ihrer URI aus der semantischen Datenbasis in der relationalen Datenbank zwischengespeichert (Abbildung 44 siehe Anhang). Die Adaptierung von Excavation-Service geschieht durch die Adaptierung von XML-SchemaTypen. Dabei werden die Arrays mit Hilfe einer For-Schleife durchlaufen und die einzelnen Elemente aus dem Input der Adapter WSDL-Schnittstelle in die Elemente XML-Schema-Typs von dem Excavation-Service umgewandelt. Diese Adaption funktioniert recht gut und basiert auf der XPath Technologie. Dabei wird jedes abzubildende Element mit Hilfe einer XPath Expression aus einem XML-Schema Typ extrahiert und in das andere Element eines anderen Typs abgespeichert. Die Verwendung von XPath kann bei komplexen Transformationen zu Performance Problemen führen, deswegen sollte in der Praxis die XSL-Transformation genutzt werden. ActiveBPEL-Engine zum Beispiel unterstützt beide Transformationsarten: XPath und XSLT.

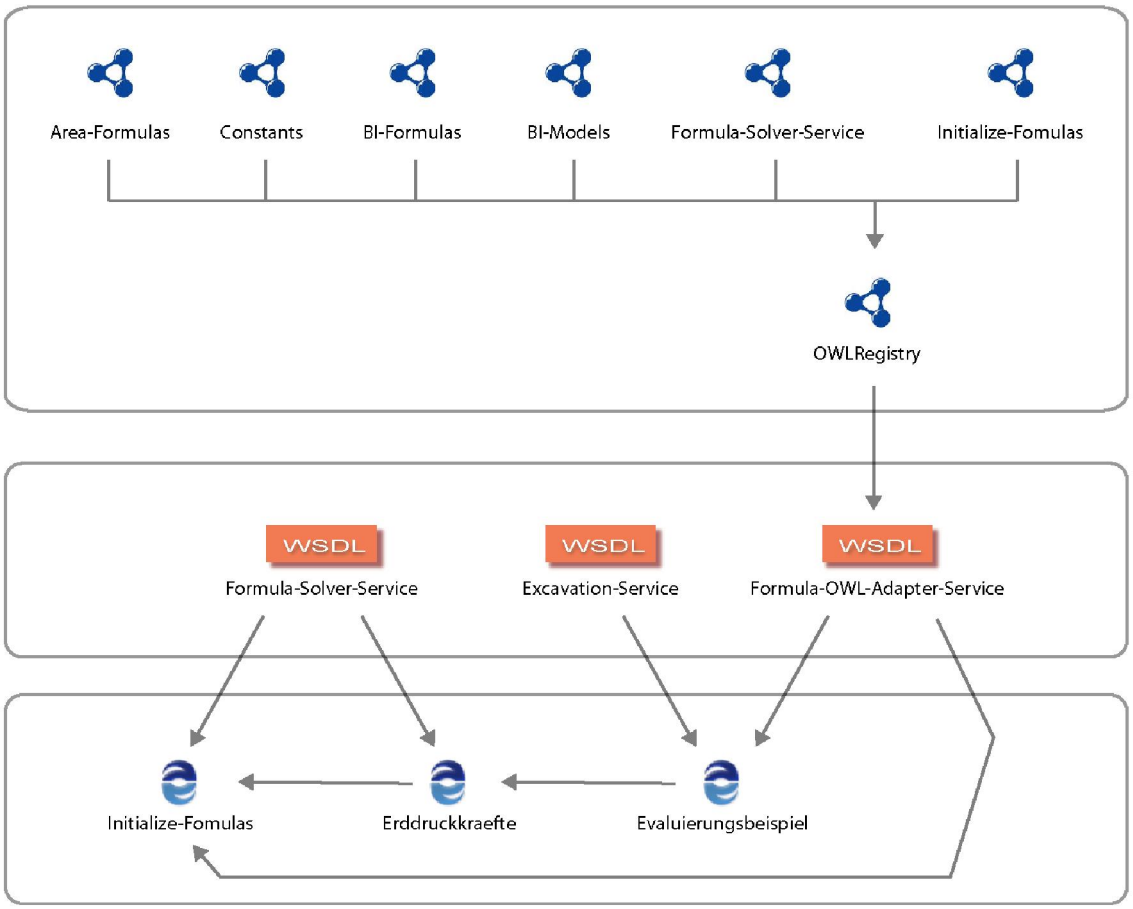

Abbildung 33 Prototyp Übersicht

Das Evaluierungsbeispiel aus dieser Arbeit besteht aus mehreren BPEL-Unterprozessen (Abbildung 45 im Anhang), orchestriert mehrere Webservices und verwendet die Formeln aus der semantischen Datenbasis. Im Anhang sind einige BPEL-Prozesse aus dem Evaluierungsbeispiel dargestellt, so ist in der Abbildung 45 der gesamte Evaluierungsprozess dargestellt. Dieser enthält viele weitere Unterprozesse, zum Beispiel den *Erddruckkräfte-*Unterprozess (Abbildung 46 im Anhang). Der Unterprozess Erddruckkräfte enthält weitere Unterprozesse, zum Beispiel einen Prozess der Vertikalspannungen berechnen kann (siehe Abbildung 47 im Anhang). Die Komplexität des Evaluierungsbeispiels zeigt, dass schon die Berechnung eines relativ „kleinen“ akademischen Beispiels zu einer relativ umfangreichen Anwendung geführt hat. Damit ein Bauingenieur ohne spezielle Informatikausbildung die Workflow Prozesse implementieren kann, müssen noch viele Tools verfeinert oder sogar neu entwickelt werden. OSPP vereinfacht die Entwicklung eines Prozesses, indem sie viele Aspekte einer Architektursprache transparent für einen Entwickler im Hintergrund verwaltet. So muss sich beispielsweise ein Programmierer nicht mehr um die Partnerlinks Typen kümmern, sondern kann einfach die Adresse und Operation des aufzurufenden Webservices angeben. Im Vergleich dazu ist die während dieser Arbeit verwendete Entwicklungsumgebung (ActiveBPEL Designer) nur für eine erfahrene Fachkraft geeignet, die sich mit der Theorie der Architektursprache auskennt und die XML-Technologien, wie X-Path, XSLT-Transformation, sowie Webservice-Standards wie WSDL, SOAP, BPEL beherrscht. Ein Bauingenieur der sich mit dem Berechnen von Bauwerken auskennt ist kein geeigneter ActiveBPEL Designer Nutzer.

Während dieser Arbeit wurde zwar OSPP nicht genutzt, weil diese noch entwickelt wird und zum Zeitpunkt dieser Diplomarbeit noch keine komplexen XML-Schema Typen unterstützte. Jedoch sollte OSPP in einer weiteren Arbeit als eine mögliche Entwicklungsumgebung für Workflow Prozesse berücksichtig werden. Denn OSPP ist viel einfacher als beispielsweise ActiveBPEL Designer zu nutzen. Außerdem wurden die neuen Werkzeuge für semantische Suche von Ressourcen (Webservices, Formeln, Dateien) prototypisch während dieser Arbeit in OSPP implementiert und sollen in einer weiterführenden Arbeit weiterentwickelt werden. Diese Werkzeuge passen OSPP an die Bedürfnisse eines Bauingenieurs an und ermöglichen durch die semantische Suche eine bessere Ressourcenverwaltung.

3.6 OSPP Erweiterung

Facettenbasierte Klassifikation ist im WWW sehr verbreitet, besonders auf kommerziellen Webseiten. Die Facetten können in einer speziell dafür entwickelten Sprache XFML (Exchangeable Facetted Metadata Language) entwickelt werden (Van Dijck, 2003). XFML basiert auf XML und ermöglicht Interoperabilität zwischen verschiedenen Systemen, sowie eine einfache Änderung von Facetten. In dieser Arbeit wurde Jedoch OWL-DL für die Definition von Facetten benutzt. OWL hat viele Vorteile gegenüber XML (siehe Abschnitt 1.3), deswegen wurde für diese Arbeit ein kleines OWL Framework entwickelt, das ähnlich wie XFML die Facetten beschreiben kann. In dieser Arbeit wurde die OSPP um zwei neue Funktionen der semantischen Suche erweitert. Die erste Erweiterung soll eine semantische Suche mit Hilfe von Facettenbasierten Klassifikation von Webservices in OSPP ermöglichen. Diese Erweiterung sollte eine effiziente Suche von benötigten Webservices für einen Workflow Prozess unterstützen und dadurch die Erstellung eines Prozesses beschleunigen, sowie die Wiederverwendung von einzelnen Webservices verbessern. Die zweite Erweiterung erweitert die OSPP um die semantische Suche von Dateien. Die Suchdialoge für Webservices und Dateien sehen gleich aus, jedoch verwenden sie jeweils eine unterschiedliche Suchlogik, die mit Hilfe eines Strategie-Entwurfsmusters (Gamma, et al., 2001) von der Gesamtlogik entkoppelt ist. Die Suchlogik ist in einem Webservice implementiert und wird aus OSPP mit Hilfe eines Clients verwendet. In der Abbildung 34 kommuniziert OSPP mit zwei Webservices, die die Suchanfragen von OSPP verarbeiten und die benötigten Ressourcen in der semantischen Datenbasis suchen. Der *OWL-S-Adapter-Service* sucht beispielsweise nach Webservices, deren semantische Beschreibungen im OWL-S Format in der semantischen Datenbasis vorliegen. Der *File-Adapter-Service* sucht nach Dateien, deren semantische Beschreibungen ebenfalls in der semantischen Datenbasis vorliegen. So kann ein Bauingenieur eine Autocad Zeichnung oder eine andere Datei suchen, die er als Input in einem Webservice braucht, den er in dem Workflow Prozess eingebaut hat. Die Suchanfragen können in OSPP mit Hilfe der Benutzeroberfläche formuliert werden. In der Abbildung 35 ist der Suchdialog für die semantische Suche von in dem Workflow benötigten Dateien dargestellt. Auf der linken Seite des Suchdialogs ist die Suchontologie für die semantische Suche von Dateien zu finden. Diese wird aus der semantischen Datenbasis geladen (siehe Abbildung 34) und kann zur Laufzeit neugeladen werden. Die Suchergebnisse werden im rechten oberen Fenster angezeigt. Wenn die gefundenen Ergebnisse angeklickt werden, werden diese in Form von XML im rechten unteren Fenster angezeigt. In einer weitführenden Arbeit sollte eine benutzerfreundliche Visualisierung von gefundenen Ressourcen wie Dateien oder Webservices entwickelt werden. Die Abbildung 35 veranschaulicht beispielsweise eine erfolgreiche Suche einer 2D-Zeichnung einer Brücke.

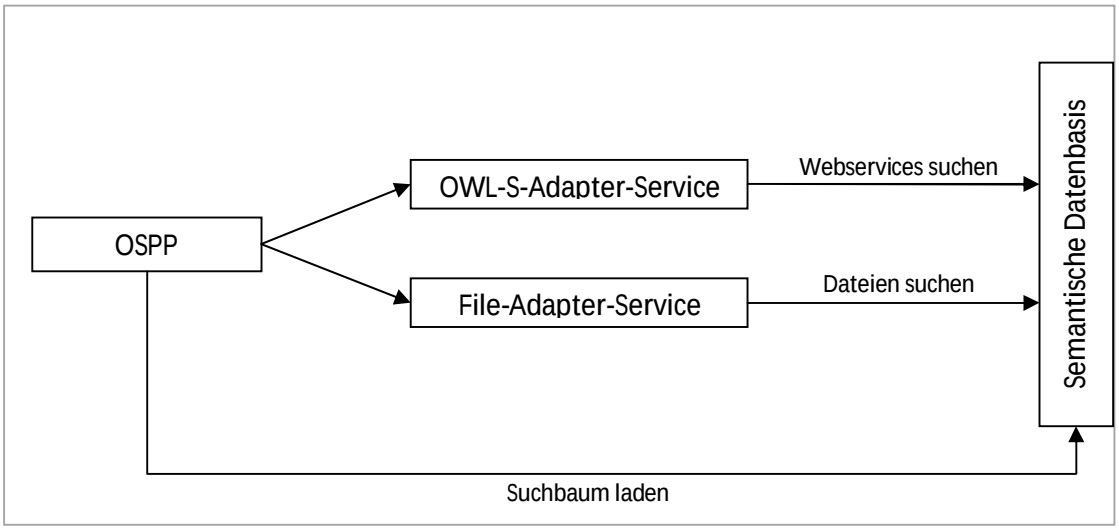

Abbildung 34 Kommunikation von OSPP mit der semantischen Datenbasis

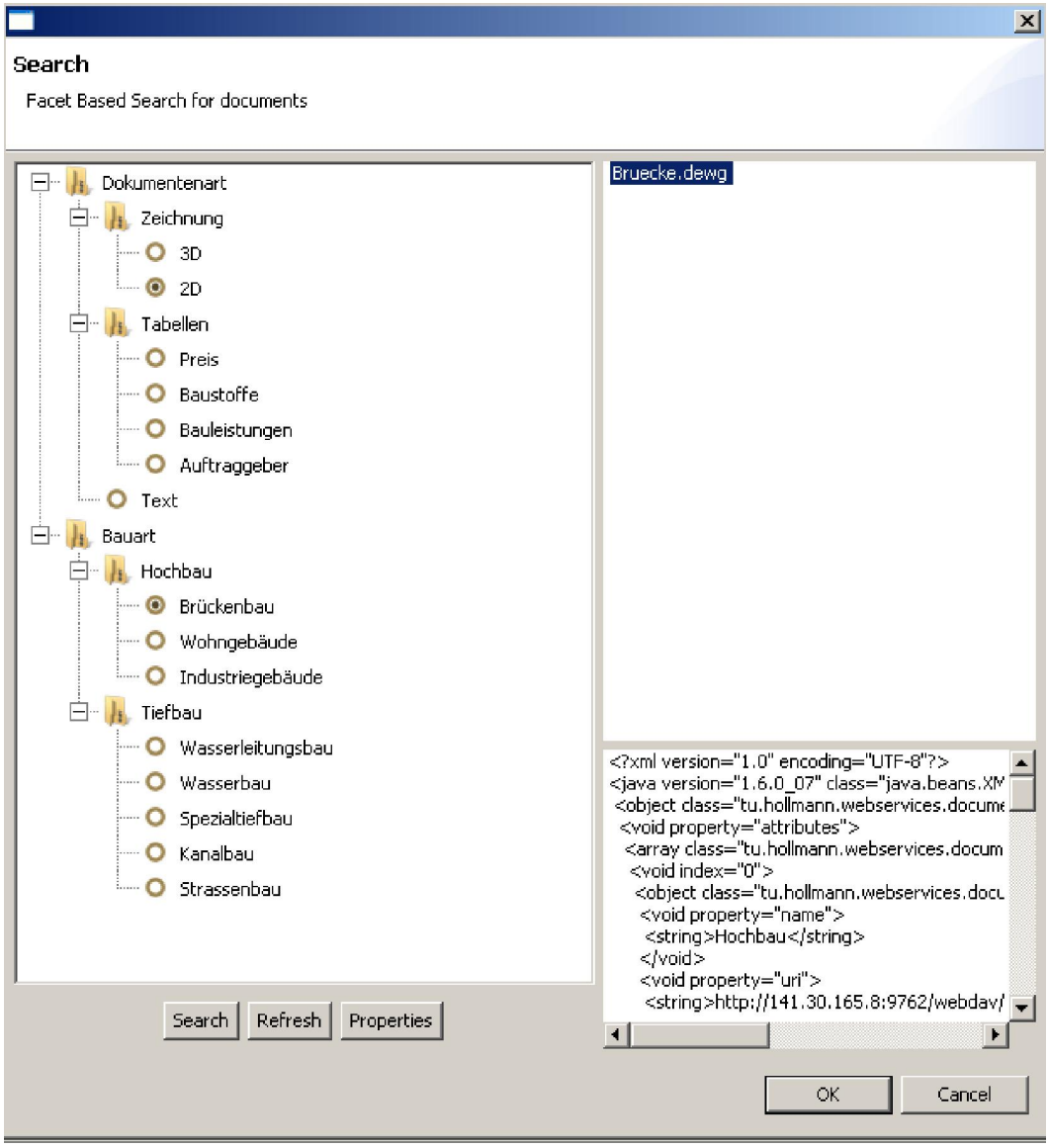

Abbildung 35 Dateiensuche in OSPP

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.6.0_07" class="java.beans.XMLDecoder">
 <object class="tu.hollmann.webservices.document.bean.xsd.Document">
  <void property="attributes">
   <array class="tu.hollmann.webservices.document.bean.xsd.Attribute"
length="4">
    <void index="0">
     <object class="tu.hollmann.webservices.document.bean.xsd.Attribute">
      ...
    <void index="2">
     <object class="tu.hollmann.webservices.document.bean.xsd.Attribute">
      <void property="name">
       <string>2D</string>
      </void>
      <void property="uri">

<string>http://141.30.165.8:9762/webdav/Documents/description/DocumentDescript
ion.owl#_2D</string>
      </void>
      <void property="url">
       <string>http://141.30.165.8:9762/webdav/OWLRegistry.owl</string>
      </void>
     </object>
    </void>
    ...
     </object>
    </void>
   </array>
  </void>
  <void property="documentURL">

<string>http://141.30.165.8:9762/webdav/Documents/data/Zeichnungen/Bruecke.dew
g</string>
  </void>
  <void property="name">
   <string>Bruecke.dewg</string>
  </void>
  <void property="uri">

<string>http://141.30.165.8:9762/webdav/Documents/description/DocumentDescript
ion.owl#Bruecke.dewg</string>
  </void>
  <void property="url">
   <string>http://141.30.165.8:9762/webdav/OWLRegistry.owl</string>
  </void>
 </object>
</java>
```

Abbildung 36 XML-Daten von der gefundenen Datei

3.7 Standards und Entwicklungswerkzeuge

Die während dieser Arbeit verwendeten Tools sind ausschließlich frei nutzbar oder sogar Open Source und stehen unter Apache-, GPL-, EPL-Lizenzen für die Nutzung bereit. In diesem Abschnitt wird diskutiert, welche Technologien und Tools für eine konkrete praxisrelevante Implementierung nötig sind. Die während dieser Arbeit verwendeten Technologien basieren auf aktuellen Standards und werden bereits heute in der Praxis oft eingesetzt. Die klassischen Standards wie WSDL, SOAP, BPEL haben sich schon seit Jahren bewehrt und werden von vielen Werkzeugen recht gut unterstützt. Während die klassischen Standards und Entwicklungswerkzeuge fast zur „alten" Welt gehören und in der Praxis täglich mit bewehrten Methoden genutzt werden, stecken die semantischen Standards und Technologien erst in „Kinderschuhen". So gibt es zwar mehrere Entwicklungswerkzeuge, die aktuelle Standards zum Teil unterstützen, jedoch haben diese noch viele Probleme nicht gelöst und ändern sich in jeder neuen Version bis in die Unkenntlichkeit, so dass sogar ein erfahrener Nutzer sich wieder einarbeiten muss um die neuen Version der Werkzeuge zu nutzen. Ein gutes Beispiel bietet der Vergleich von Protegé (Sta08) in der Version 3.x und 4.x. Ein Grund für ständige Änderung der Entwicklungswerkzeuge wie Protegé ist, dass für die Visualisierung von semantischen Inhalten noch keine gemeinsamen Richtlinien existieren. Es gibt allein für Protegé mehr als ein Dutzend verschiedener Visualisierungswerkzeuge. Im Gegensatz zur Präsentation von semantischen Inhalten hat sich die Logik für das Verarbeiten von semantischen Informationen schon heute in vielen Bereichen bewährt und wird beispielsweise durch OWL-Teilsprachen standardisiert. Für die Lösung der in dieser Arbeit gestellten Aufgaben wurde das Jena-Framework (Jen08) wegen der Unterstützung von SPARQL genutzt. Das Jena-Framework kann über die DIG-Schnittstelle (DIG08) die Inferenzmaschinen von Drittanbietern ansprechen. Derzeit gibt es viele Inferenzmaschinen, die bekanntesten sind Pellet(pel08), RacerPro(Rac08), FaCT++(FaC08). In dieser Arbeit wurde Pellet genutzt. Eine Inferenzmaschine kann als ein Server laufen und beispielsweise aus einem Webservice über DIG-Schnittstelle (DIG08) verwendet werden. Als J2EE Server für verschiedene Webanwendungen wird in dieser Arbeit der Tomcat Server verwendet. Tomcat implementiert offiziell den Java Servlet und Java Server Pages Standard Spezifikation von Sun Microsystems. Das ist geschichtlich bedingt, denn Apache Tomcat ist eine Weiterentwicklung von dem Java Web Server und JServ. Java Web Server war der erste Servlet Container, der die Möglichkeiten von J2EE Technologie demonstrierte. JServ war ein J2EE-Modul für Apache Server von Apache Fundation. Sun hat den Source Code von dem Java Web Server an Apache Fundation übergeben und Tomcat 3.x war eine Verschmelzung von JServ und Java Web Server (Vivek Chopra, 2004). Jede Tomcat Version implementiert eine bestimmte Servlet und JSP Spezifikation. So implementiert der in dieser Arbeit verwendete Tomcat 5.5 die Servlet 2.4 und JSP 2.0 Spezifikation (Tomcat). Der Tomcat implementiert nicht vollständig J2EE Standard, zum Beispiel implementiert er nicht die Enterprice Java Bean Spezifikation. Für eine prototypische Anwendung aus dieser Arbeit ist der Tomcat ausreichend, der lief stabil lange Zeit und stürzte manchmal wegen ActiveBPEL Engine ab, weil diese viele BPEL-Unterprozesse aufgerufen und sehr viel Arbeitsspeicher verbraucht hat. Ein Server sollte solche Fehler tolerieren und lieber die Webanwendung, die diese Fehler verursacht abschalten, als vollständig wegen unzureichendem Arbeitsspeicher abzustürzen und dadurch auch alle anderen laufenden Webanwendungen abzuschalten. Für eine Praxisorientierte Anwendung sollte eine andere Konfiguration als Tomcat 5.5 und ActiveBPEL Egine 4.1 verwendet werden. Der Tomcat Server wurde in dieser Arbeit hauptsächlich wegen ActiveBPEL Engine (The08) verwendet. Diese läuft in dem Tomcat als eine Webapplikation und arbeitet sehr eng mit der ebenfalls von ActiveBPEL entwickelten frei verwendbaren Entwicklungsumgebung (ActiveBPEL Designer

(Act08)) für die Erstellung von BPEL-Prozessen zusammen. Beide Tools haben zwar einige Mängel, jedoch waren diese für eine prototypische Implementierung des in Abschnitt 2.2 beschriebenen und dieser Arbeit zugrunde liegenden Konzepts ausreichend. Sie unterstützen mehrere Workflow Standards wie WS-BPEL 2.0, BPEL4People und WS-Human, außerdem bieten sie die Werkzeuge für Debuging, Simulation, Testing, Deploying, Logging an.

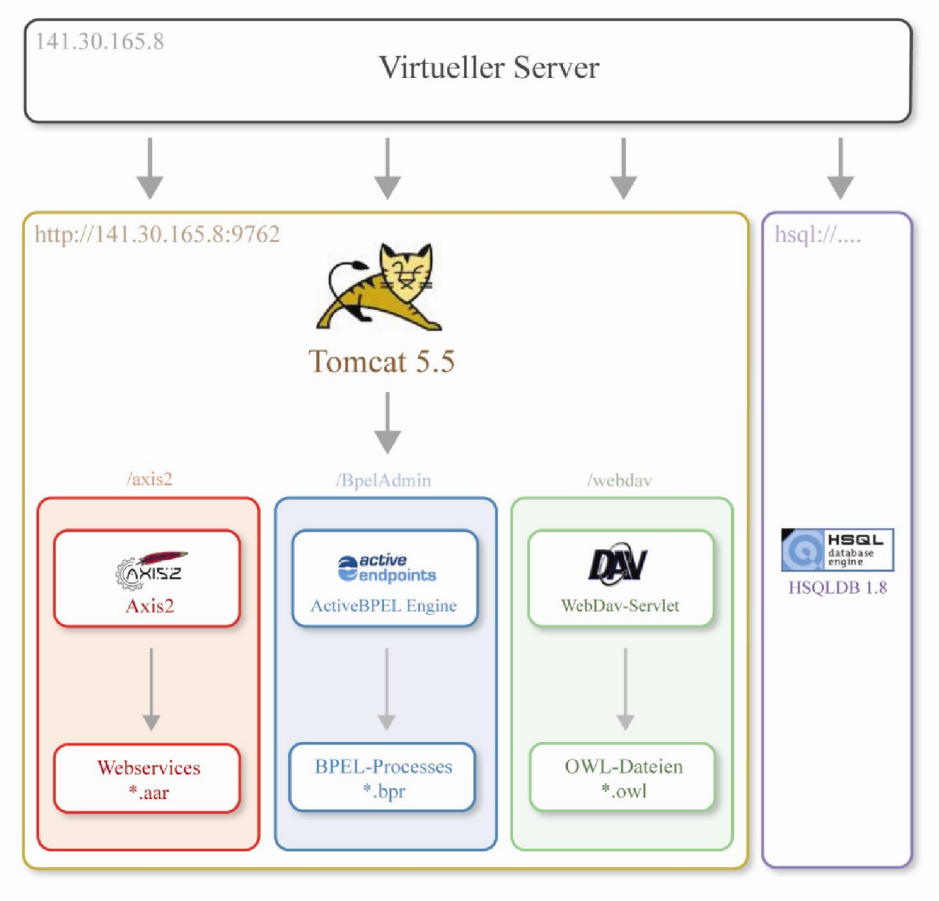

Abbildung 37 Entwicklungsumgebung - Virtueller Server

Die in einem Workflow Prozess verwendeten Daten sollen nach dem Konzept aus der semantischen Datenbasis geladen werden. Die semantische Datenbasis wurde aus mehreren während dieser Arbeit entstandenen und im WWW veröffentlichten OWL-Dokumenten zusammengestellt. Für die Veröffentlichung von OWL-Dokumenten im WWW wurde im Tomcat das standardmäßig mitgelieferte WebDav Servlet verwendet. In der Praxis muss dabei das Problem der Sicherheit gelöst werden, damit nur die autorisierten Nutzer auf die in der semantischen Datenbasis veröffentlichten Informationen zugreifen dürfen. Die Konzepte sowie die Forschungsziele für die Sicherheit im Semantic Web werden in (Bauer, 2008) umfassend vorgestellt. Für die Zwischenspeicherung von mathematischen Gleichungen wurde eine relationale Datenbank benötigt. Zum Einsatz kam die HSQLDB (hsq08). Diese wurde über das Hibernate Framework (Hibernate, 2008) verwaltet und kann deswegen gegen eine andere von dem Hibernate unterstützte Datenbank ausgetauscht werden, zum Beispiel Apache Derby (Apa081). Tomcat wurde auf einem virtuellen Server mit einer statischen IP installiert (siehe Abbildung 37), dadurch konnten die in Axis2 veröffentlichten Webservices in BPEL-Prozessen durch statische Deployment Descriptoren bereitgestellt werden. Die WSDL-Dokumente von solchen Webservices konnten unter einer festen URL gefunden und in einer OWL-S Beschreibung eines Webservices referenziert werden. Die OWL-S Beschreibungen, sowie auch

andere OWL-Dokumente konnten unter statischen URLs im WWW veröffentlicht und durch URIs aus anderen OWL-Dokumenten referenziert werden. Die mit Hilfe von OWL-S semantisch beschriebenen Webservices wurden mit Hilfe von Axis entwickelt. Axis2 ist ein Open Source Projekt von Apache Software Fundation und ist ein Nachfolger von Axis. Im Gegensatz zu Axis ist Axis2 nicht mehr RPC/Encoded, sondern Dokument orientiert. Im Gegensatz zum RCP-unterstützt Dokument Style die Validierung des High-Level Business Dokuments als ein Aspekt der Webservice Engine. RCP/Encoded Style muss die Validierung bei Bedarf in der Methode selbst implementieren, was bei größerer Software zu Problemen führen kann. Ein guter Vergleich zwischen RCP/Ecoded und Dokument-Style kann in (McCarthy, 2002) nachgelesen werden. Außerdem ist Axis2 schneller, verbraucht weniger Ressourcen und ermöglicht viele Kommunikations-Szenarien, die in Axis nur schwer oder gar nicht realisierbar waren (Frotscher, et al., 2007). In dieser Arbeit läuft Axis2 als eine Webapplikation in einem Servlet Container.

4. Zusammenfassung

Schon heute sind für kontinuierliche Überwachungen von Bauwerken Einzelfalllösungen im Einsatz. Die Anwendungs- und Datenintegration, die die flexible Einbindung von Ressourcen in den Überwachungsworkflow ermöglicht, ist jedoch bisher in der Baupraxis noch nicht gelöst. Aktuelle bekannte Probleme während des Monitoringprozesses lassen die Schlussfolgerung zu, dass für die Vorhersage des künftigen Verhaltens des Systems sowohl die Anpassung der Parameter als auch die Wahl des geeigneten Modells notwendig sind. Das Austauschen und die gezielte Auswahl des benötigten Modells wird jedoch durch bisherige Monitoringinformationsmanagementsysteme kaum unterstützt (Scherer, et al., 2007). In dieser Arbeit wurde ein Prototyp für ein verteiltes, komponentenbasiertes Informationssystem für die Bauwerksüberwachung entwickelt. Ein Evaluierungsbeispiel zeigt die Vorteile der Austauschbarkeit von Auswertesoftware sowie die neuen Möglichkeiten, die sich durch eine verteilte semantische Datenbasis für die Variation und den Austausch von Ingenieurmodellen ergeben, auf. Diese Arbeit hat gezeigt dass es praktisch durchaus möglich ist ein Informationssystem zur Bauwerksbeobachtung durch flexible Kombination von Web-Ressourcen zu erstellen. Das Informationssystem basiert auf semantischer Beschreibung von Sensordaten, Modellen und Workflow Prozessen. Das Konzept wurde prototypisch mit aktuellen Tools umgesetzt und kann in der weiterführenden Arbeit zu einem neuartigen Informationssystem ausgebaut werden. Dabei sollte die OSPP als eine moderne, erweiterbare Workflowumgebung als eine mögliche Plattform für dieses Informationssystem dienen. In der OSPP können die nötigen Ressourcen für das Erstellen eines Workflow Prozess semantisch mit Hilfe von Facetten gesucht werden. Somit ermöglicht OSPP eine effizientere Erstellung von Workflow Prozessen und ermöglicht eine bessere Zusammenarbeit und eine bessere Wiederverwendung.

Während dieser Arbeit wurden einfache Berechnungsmodelle, die durch gewöhnliche mathematische Formeln definierbar sind, genommen und prototypisch in BPEL-Prozessen für die Berechnung der Horizontallverschiebung einer Spundwand genutzt (siehe Abschnitt 3.5). In einer weiterführenden Arbeit sollte eine Möglichkeit zum Speichern und Ausführen von

komplexen Berechnungsmodellen, die im OWL-Format definiert sind, erforscht werden. Das Austauschen von Berechnungsmodellen würde einem Bauingenieur ermöglichen, komplexe Berechnungen mit geringerem Aufwand zu lösen. Die OSPP sollte dabei eine wichtige Rolle spielen, denn sie vereinfacht das Deployment eines an die neue Problematik zur Laufzeit angepassten Prozesses. Da ein Teil der Logik in dem Berechnungsmodell ausgelagert werden kann, sollte eine Methode entwickelt werden, die die Suche eines Berechnungswebservices mit der Suche eines Berechnungsmodells kombiniert. Die semantische Datenbasis sollte um Zugriffsrechte erweitert werden. Eine gute Übersicht über die Sicherheit im Semantic Web bietet (Bauer, 2008). Ebenso sollte eine Möglichkeit erarbeitet werden, die einem Nutzer eine semantische Verwaltung von Ressourcen wie Webservices und Dateien ermöglicht. Die Formulierung von Suchanfragen, sowie die Präsentation der Suchergebnisse durch eine benutzerfreundlichere Benutzeroberfläche sollte mehr erforscht und gegebenfalls verbessert werden. Dabei sollten die Entwicklungen aus der Mediengestaltung und Psychologie berücksichtigt werden. Die Benutzeroberfläche sollte an die gesuchten Ressourcen angepasst werden, so sollte sich die Präsentation eines Webservices sich von der Präsentation einer Datei unterscheiden und einem Nutzer die Informationen über die gefundenen Ressourcen besser präsentieren.

5. **Anhang**

5.1 Abbildungen

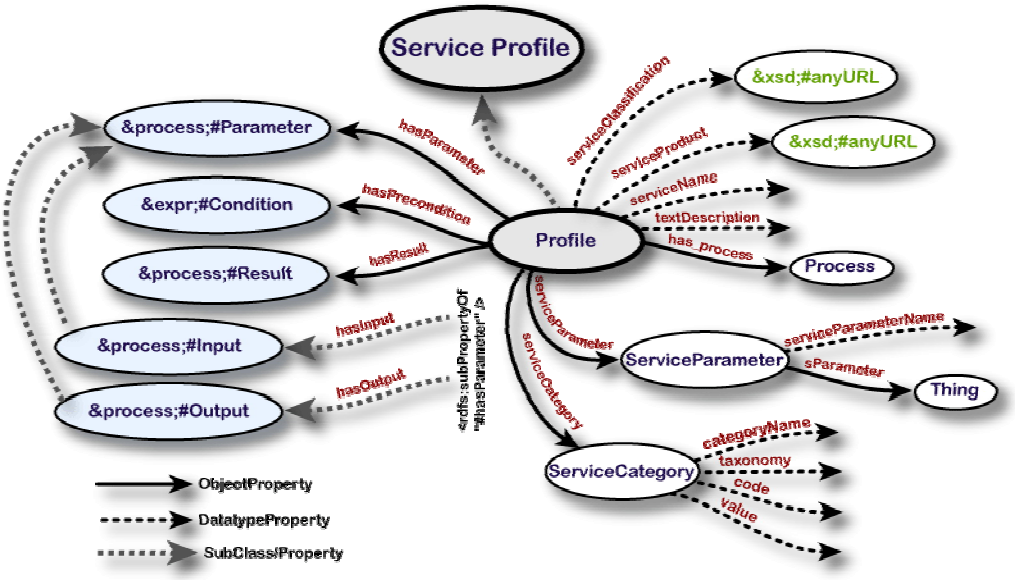

Abbildung 38 Ausgewählte Klassen und Eigenschaften von dem Profile aus dem OWL-S Standard (Martin, et al., 2004)

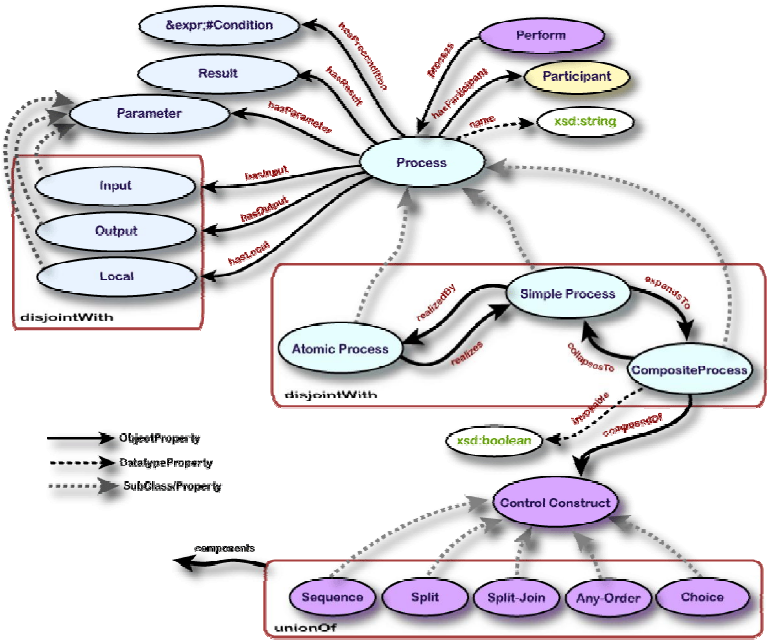

Abbildung 39 Top-Level von der Prozess Ontologie aus dem OWL-S Standard (Martin, et al., 2004)

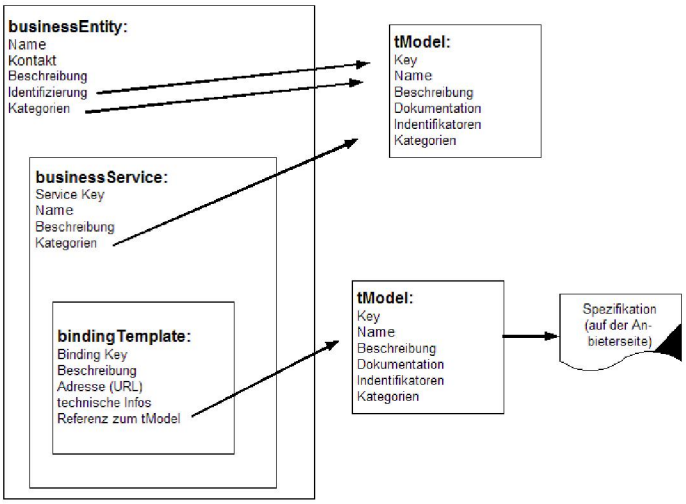

Abbildung 40 Zusammenhang zwischen Datenstrukturen einer UDDI-Registry (Web081)

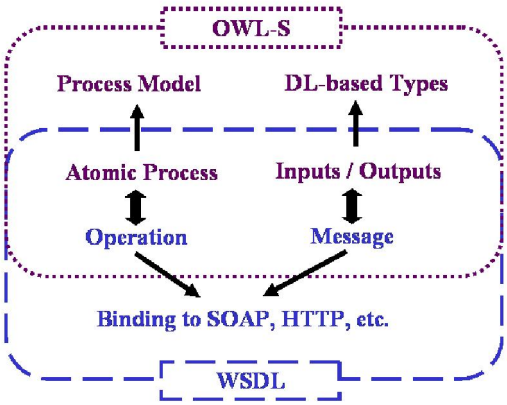

Abbildung 41 Mapping zwischen dem OWL-S Standard und dem WSDL-Standard (Martin, et al., 2004)

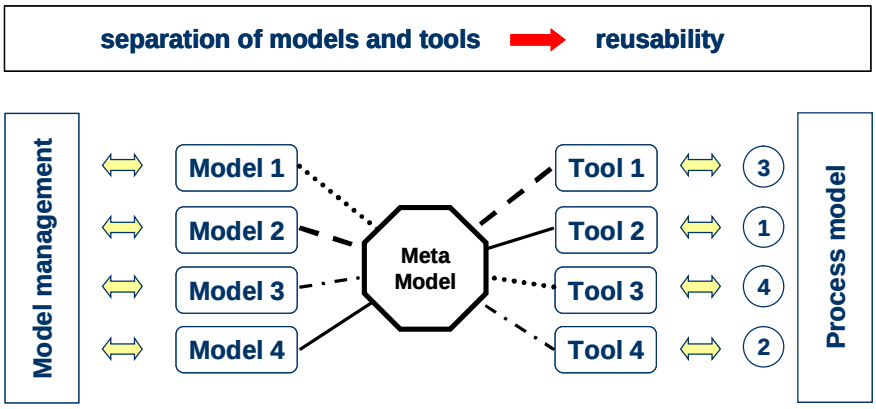

Abbildung 42 Komponentenbasierter Ansatz für die Wiederverwendung von Modellen. Dieser Ansatz bildet die Grundlage für das während dieser Arbeit entwickelten Monitoringsystems (Faschingbauer, et al., 2007)

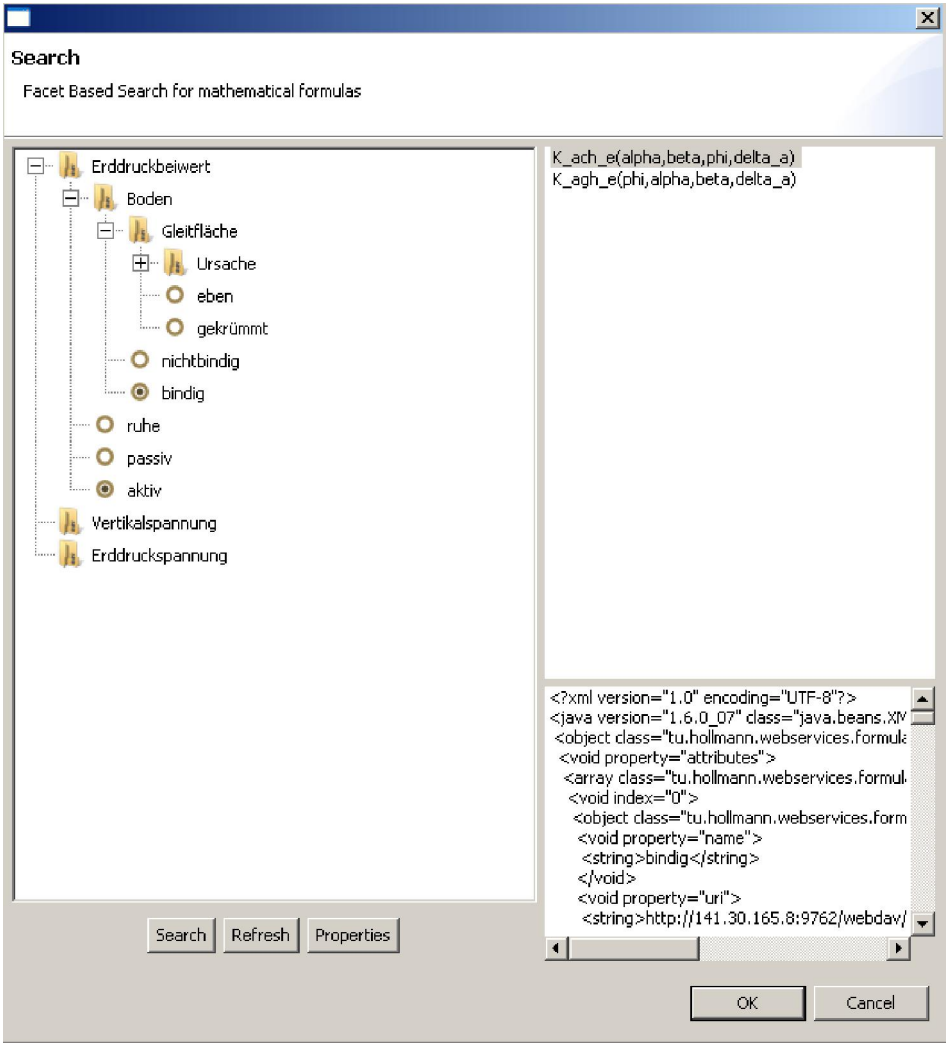

Abbildung 43 Benutzerdialog aus OSPP für die Suche nach mathematischen Gleichungen

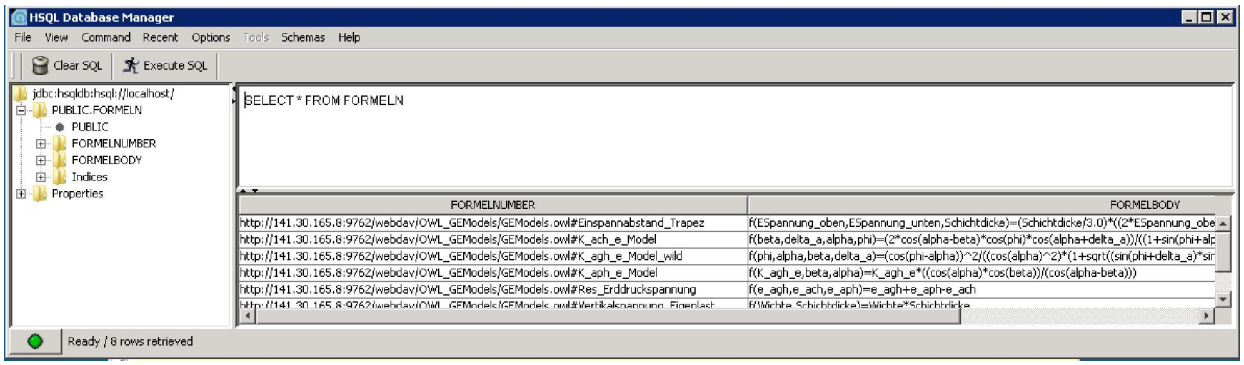

Abbildung 44 Die von dem Formula-Solver-Service in einer relationalen Datenbank zwischengespeicherten Formeln

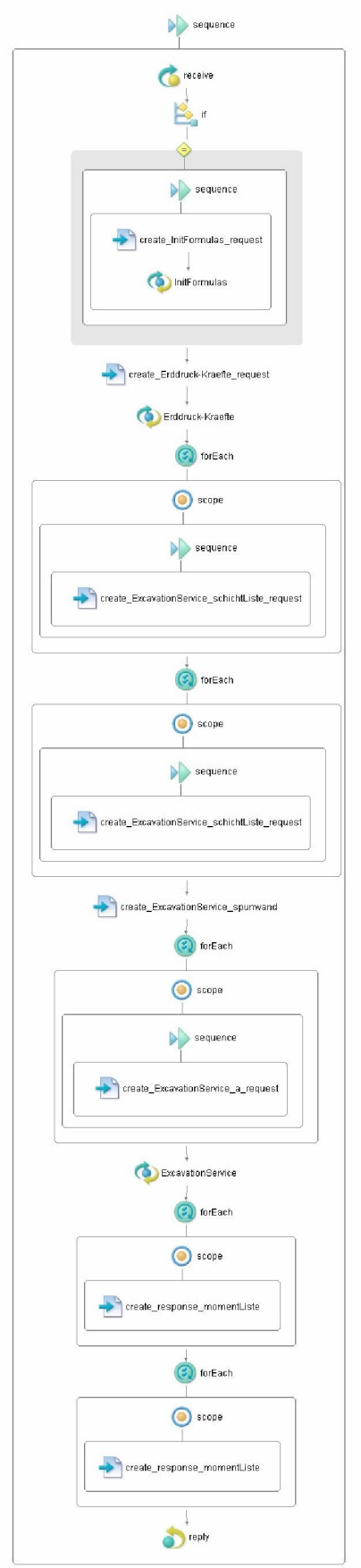

Abbildung 45 BPEL-Prozess Evaluierungsbeispiel

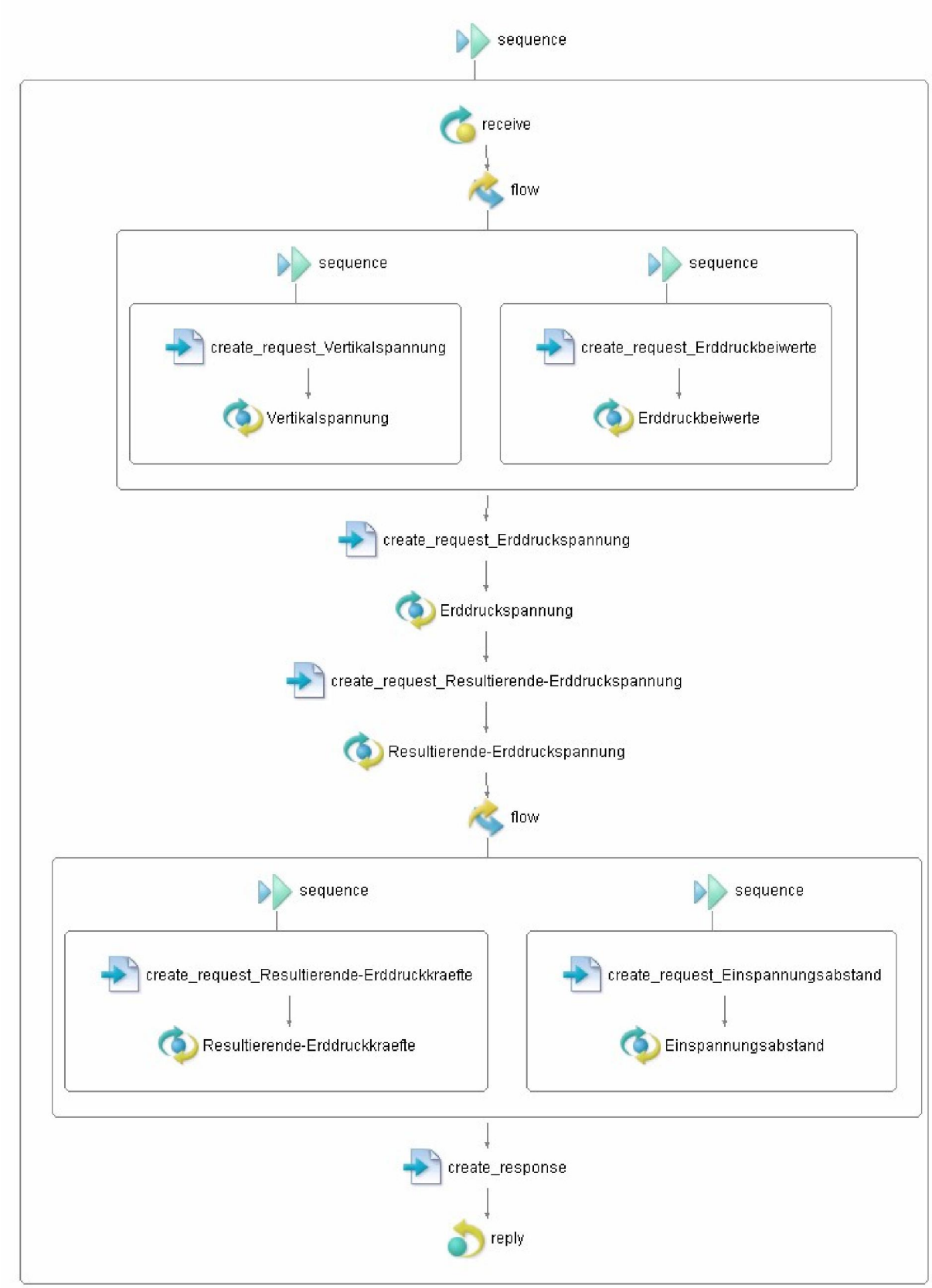

Abbildung 46 BPEL-Prozess Erddruckkräfte. Dieser Prozess wird aus dem BPEL-Prozess Evaluierungsbeispiel als ein Unterprozess aufgerufen.

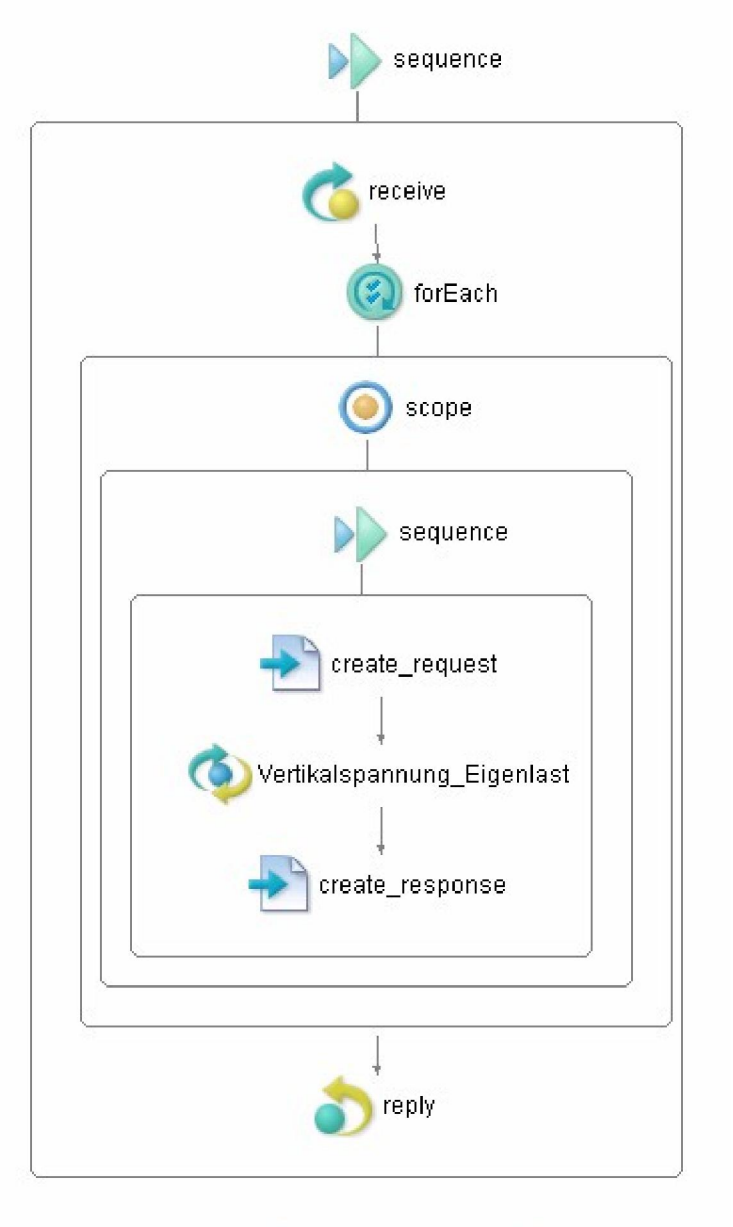

Abbildung 47 BPEL-Prozess Vertikalspannung wird aus dem BPEL-Prozess Erddruckkräfte als ein Unterprozess aufgerufen.

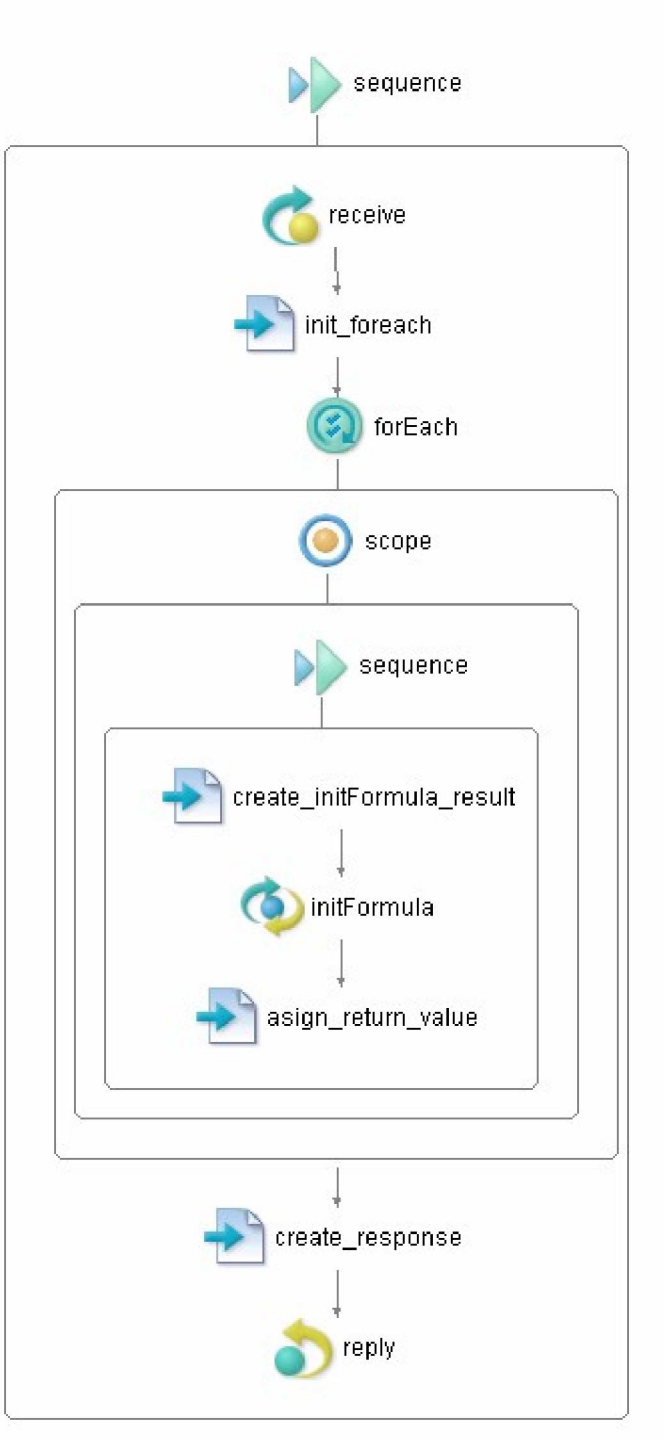

Abbildung 48 BPEL-Prozess initFormulas. Dieser Prozess extrachiert viele Formeln aus der semantischen Datenbasis und speichert diese mit Hilfe von Formula-Solver-Servcie in der relationalen Datenbasis.

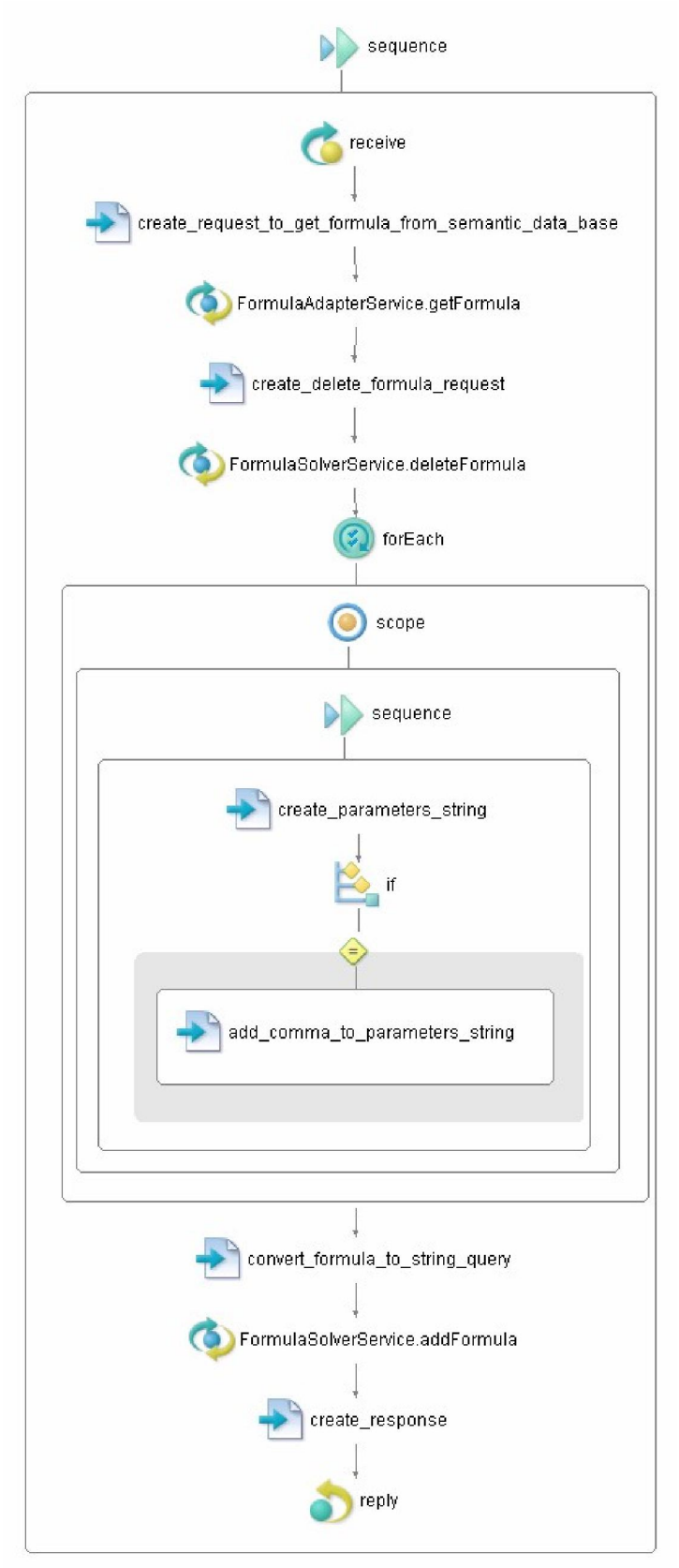

Abbildung 49 BPEL-Prozess initFormula wird als ein unterprozess aus dem initFormulas-Prozess aufgerufen.

5.2 Tabellenverzeichnis

Funktionale-Services	
Formula-Solver-Service	berechnet die mathematischen Gleichungen
Exavation-Service	berechnet die Horizontalverschiebung
Adapter-Services	
Formula-OWL-Adapter-Service	semantische Suche und Extraktion in einen XML-Schema Typ von Formeln aus der semantischen Datenbasis
File-Adapter-Service	semantische Suche von Dateien aus der semantischen Datenbasis
OWL-S-Adapter-Service	semantische Suche von Webservices in der semantischen Datenbasis
SPARQL-Service	führt die SPARQL-Anfragen auf einem RDF-Dokument aus.

Tabelle 2 Diese Tabellebietet eine Übersicht von während dieser Arbeit implementierten Webservices

Ressourcen-Arten in der semantischen Datenbasis	
Formeln	Mathematische Gleichungen werden mit Hilfe eines dafür entwickelten OWL-Frameworks beschrieben, dabei sind die Daten in OWL-enthalten.
Dateien-Beschreibungen	Die Dateien werden mit Hilfe von OWL beschrieben, die Daten sind in Dateien enthalten.
Webservices-Beschreibungen	Webservices werden mit Hilfe von OWL-S Standards beschrieben.
Suchontologie	Das Suchbaum für semantische Suche in OSPP, wird ebenfalls mit Hilfe eines OWL-Frameworks beschrieben. Die Individuen die einen Suchbaum beschreiben werden ebenfalls in der semantischen Datenbasis gespeichert.

Tabelle 3 Eine Übersicht über die in der semantischen Datenbasis enthaltenden Informationsarten

5.3 Abbildungsverzeichnis

5.4 Literaturverzeichnis

Web Service Technologien: UDDI, SAML [Online]. - FIN, IVS, AG Softwaretechnik. - 9. 9 2008. - http://ivs.cs.uni-magdeburg.de/~dumke/WS/WS-06.html.

ActiveBPEL™ Designer [Online]. - 5. 4 2008. - http://www.active-endpoints.com/active-bpel-designer.htm.

Allenmang Dean und Hendler James Semantic Web for the Working Ontologist [Buch]. - Burlington : Morgan Kaufmann Publishers, 2008. - 978-0-12-373556-0.

Alves Alexandre [et al.] Web Services Business Process Execution Language Version 2.0 [Online] // docs.oasis-open.org. - OASIS, 4 11, 2007. - 9 9, 2008. - http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

Apache Axis2 - Web services engine [Online]. - 12. 8 2008. - http://ws.apache.org/axis2/.

Apache Derby [Online]. - 6. 7 2008. - http://db.apache.org/derby/.

Apache Tomcat - J2EE Servlet Container [Online]. - 2008. 8 12. - http://tomcat.apache.org/.

Bauer Bernhard Konzepte und Techniken für das Semantic Web [Online]. - 2008. - 12. 9 2008. - http://www.informatik.uni-augsburg.de/lehrstuehle/swt/vs/lehre/WS_07_08/seminar_semantics/.

Bayer Thomas REST Web Services [Online] // Orientation in Objects. - Orientation in Objects GmbH, 11 2002. - 4. 9 2008. - http://www.oio.de/public/xml/rest-webservices.htm.

Bellwood Tom, Clément Luc and von Riegen Claus UDDI Version 3.0.1 [Online]. - UDDI Spec Technical Committee Specification, 10 14, 2003. - 9 2008, 9. - http://uddi.org/pubs/uddi-v3.0.1-20031014.htm.

Berners-Lee Tim Semantic Web Road map [Online] // www.w3.org. - W3C, 10 14, 1998. - 9 10, 2008. - http://www.w3.org/DesignIssues/Semantic.html.

Berners-Lee Tim, Hendler James and Lasila Ora The Semantic Web [Article] // Scientific American Magazine. - 5 17, 2001. - http://www.sciam.com/article.cfm?id=the-semantic-web.

Cardoso Jorge Semantic Web Services: Theory, Tools, and Applications [Book]. - Hershey, New York : InformatIon Sclence reference, 2007.

Cerami Ethan Web Services Essentials [Book]. - [s.l.] : O'Reilly, 2002. - Vol. First Edition. - 0-596-00224-6.

Claburn Thomas Web 2.0 Arrives to Find Web 3.0 Underway [Online] // Dr.Dobb's Portal. - 16. 8 2007. - 10. 9 2008. - http://www.ddj.com/web-development/199001127.

Denton William Wie man eine Facettenklassifikation erstellt und diese ins Web bringt / Übers. Jursa Jan. - http://iainstitute.org/de/translations/000551.html : [s.n.], 2 2007.

DIG Interface [Online]. - 9 5, 2008. - http://dig.sourceforge.net/.

Dostal Wolfgang, Jeckle Mario und Kriechbaum Werner Semantik und web services: Beschreibung von semantik [Online]. - 4 2004. - 11. 9 2008. - http://www.jeckle.de/semanticWebServices/semantik.html.

Elenius Daniel [et al.] The OWL-S Editor – A Development Tool for Semantic Web Services. - Menlo Park, California, USA : [s.n.].

Endres Erik [et al.] Reasoners for the Semantic Web (Kaon & Kaon2) // Seminar A.I. Tools. - Universität des Saarlandes : [s.n.], 2006.

Expression4J - Mathematik Expresion Parser [Online]. - 12. 8 2008. - http://expression4j.sourceforge.net/.

FaCT++ is the new generation of the well-known FaCT OWL-DL reasoner [Online]. - 8 12, 2008. - http://owl.man.ac.uk/factplusplus/.

Faschingbauer G and Scherer R.J. Model and Sensor Data Management for Geotechnical Engineering Application [Conference] // "Bringing ITC knowledge to work", Proceedings of 24th W78 Conference. - Maribor, Slovenia : [s.n.], 2007. - 978-961-248-033-2.

Faschingbauer Gerald Modell- und Sensordatenverwaltung für die Überwachung von Grundbauwerken während der Ausführungsphase [Artikel]. - Dresden : Institut für Bauinformatik, Technische Universität Dresden, 2007.

Faschingbauer Gerald und Scherer Raimar J. Model and Sensor Data Management for Geotechnical Engineering Application [Konferenz] // "Bringing ITC knowledge to work", Proceedings of 24th W78 Conference. - Maribor, Slovenia : [s.n.], 2007. - 978-961-248-033-2.

Fensel Dieter [et al.] Enabling SemanticWeb Services [Buch]. - Berlin Heidelberg : Springer-Verlag, 2007.

Frotscher Thilo, Teufel Marc und Wang Dapeng Java Web Services mit Apache Axis2 [Buch]. - http://www.entwickler-press.de : entwickler.press, 2007. - 978-3-935042-81-9.

Gamma Erich, Helm Richard und Johnson Ralph Entwurfsmuster . Elemente wiederverwendbarer objektorientierter Software [Buch]. - [s.l.] : Addison-Wesley, 2001. - 2 : S. 484. - 978-3827318626.

GattMath [Online] // The gattMath Project Page. - 5. 10 2008. - http://gattmath.sourceforge.net/.

Geihs Kurt und C. Jaeger Michael Projektbericht DAML-S/OWL-S-Matcher [Bericht]. - Fakultät IV - Institut für Telekommunikationssysteme Intelligente Netze und Management verteilter Systeme : [s.n.], 2004.

Gepting Alexander Grundlagen von MOF.

Google JenaBean - A library for persisting java beans to RDF [Online]. - 10 15, 2008. - http://code.google.com/p/jenabean/.

Habich Dirk [et al.] Open Service Process Platform 2.0 [Conference] // Proceedings of the 2008 IEEE International Conference on Services Computing. - Hawaii, USA : [s.n.], 2008.

Hibernate Hibernate - Relational Persistence for Java and .NET [Online] // Hibernate - Relational Persistence for Java and .NET. - 8 12, 2008. - 8 12, 2008. - http://hibernate.org/.

Hitzler Pascal [et al.] Semantic Web: Grundlagen [Buch]. - Berlin : Springer, 2007. - S. 277. - 978-3540339939.

Hollmann Andreas, Wülfing Alexander, Faschingbauer, Gerald und Richly Sebastian Semantische Beschreibung und Integration von Ingeniermodellen in serviceorientierte Architekturen [Paper]. - Dresden : TU-Dresden, 2008.

hsqldb - 100% Java Database [Online]. - 5. 7 2008. - http://hsqldb.org/.

http://tomcat.apache.org/index.html [Online]. - 9 1, 2008. - http://tomcat.apache.org/index.html.

Hull Richard [et al.] E-Services: A Look Behind the Curtain. [Report]. - San Diego, USA : In Proceedings of the 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), 2003.

Jaeger Michael C. [et al.] Ranked Matching for Service Descriptions using OWL-S [Book]. - Berlin : Springer Berlin Heidelberg, 2005. - 978-3-540-24473-8.

James McGovern Rahim Adatia, Yakov Fain, Jason Gordon, Ethan Henry, Walter Hurst, Ashish Jain, Mark Little, Vaidyanathan Nagarajan, Harshad Oak, Lee Anne Phillips Java™ 2 Enterprise Edition 1.4 Bible [Book]. - Indianapolis : Wiley Publishing, Inc., 2003. - 0-7645-3966-3.

Jena – A Semantic Web Framework for Java [Online]. - 9 11, 2008. - http://jena.sourceforge.net/.

Kwasnick Barbara H. The role of classification in knowledge representation and discovery. [Online]. - Library Trends, 1999. - 17. 9 2008. - http://findarticles.com/p/articles/mi_m1387/is_1_48/ai_57046525.

Manecke Hans Jürgen Klassifikation, Klassieren. In: Grundlagen der praktischen Information und Dokumentation. [Buch]. - München : Saur, 2004. - Bd. 5.

Martin David [et al.] OWL-S: Semantic Markup for Web Services [Online]. - W3C, 11 22, 2004. - 8 9, 2008. - http://www.w3.org/Submission/OWL-S/.

McCarthy James Reap the benefits of document style Web services [Artikel]. - [s.l.] : IBM, 2002.

Mias Jeré Die Klassifikation [Online]. - 5 2000. - 13. 10 2008. - http://www.jere-mias.de/biwi/klass.html#fn4.

Netcraft Web Server Survey [Online]. - heise online, 11 2006. - 5. 10 2008. - http://news.netcraft.com/archives/2006/11/01/november_2006_web_server_survey.html.

Nielsen Jakob Designing web usability: the practice of simplicity. Indianapolis, IN: New Riders [Book]. - [s.l.] : Peachpit Press, 2000. - p. 432. - 978-1562058104.

Nohr Holger Systematische Erschließung in deutschen Öffentlichen Bibliotheken/ Holger Nohr [Buch]. - Wiesbaden : Harrassowitz, 1996. - S. 140. - 3447037873.

Oberhauser Otto Automatisches Klassifizieren [Master's Thesis]. - Wien : Dr. Otto Oberhauser, 27. 7 2004.

OMG OMG's MetaObject Facility [Online]. - 8 15, 2008. - http://www.omg.org/mof/.

pellet [Online] // Pellet: The OpenSource OWL DL Reasoner. - 12. 9 2008. - http://pellet.owldl.com/.

Polowinski Jan Faceted Browsing. - Dresden : TU-Dresden, 8 18, 2008.

Protégé is a free, open source ontology editor and knowledge-base framework [Online]. - Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine.. - 5. 8 2008. - http://protege.stanford.edu/.

Prud'hommeaux Eric und Seaborne Andy SPARQL Query Language for RDF [Online]. - W3C, 15. 1 2008. - 11. 9 2008. - http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.

RacerPro [Online]. - 11. 9 2008. - http://www.racer-systems.com.

Rao Jinghai Semantic Web Service Composition via Logic-based Program Synthesis [Buch]. - Trondheim : Department of Computer and Information Science, Norwegian University of Science and Technology;, 2004. - 82-471-6463-9.

Scherer J. Reimar und Faschingbauer Gerald Überwachung komplexer Ingenieur-strukturen auf der Basis von Informations- und Kommunikationstechnologien [Artikel]. - Dresden : Institut für Bauinformatik, TU-Dresden, 2007.

SOAP Version 1.2 Part 0: Primer (Second Edition) [Online]. - W3C. - 4. 9 2008. - http://www.w3.org/TR/2007/REC-soap12-part0-20070427/.

Srinivasan Naveen, Paolucci Massimo und Sycara Katia An Efficient Algorithm for OWL-S Based Semantic Search in UDDI [Buch]. - Robotics Institute, Carnegie Mellon University, USA : Springer Berlin / Heidelberg, 2005. - 978-3-540-24328-1.

The ActiveBPEL Community Edition Engine [Online]. - 12. 8 2008. - http://www.activevos.com/community-open-source.php.

The OWL API [Online]. - 7. 8 2008. - http://owlapi.sourceforge.net/.

The OWL API [Online]. - 7 18, 2008. - http://owlapi.sourceforge.net/.

The OWL-S Editor [Online]. - 26. 10 2008. - http://owlseditor.semwebcentral.org/documentation.shtml.

Van Dijck Peter XFML Core - eXchangeable Faceted Metadata Language [Online]. - 2 14, 2003. - 9 18, 2009. - http://petervandijck.com/xfml/spec/1.0.html.

Vasiliev Yuli SOA and WS-BPEL [Book]. - Olton : Packt Publishing Ltd., 2007. - 978-1-847192-70-7.

Vivek Chopra Amit Bakore, Jon Eaves, Ben Galbraith, Sing Li, Chanoch Wiggers Professional Apache Tomcat 5 [Book]. - Indianapolis : Wiley Publishing, Inc., 2004.

WebDAV-Servlet [Online]. - 12. 8 2008. - http://webdav-servlet.sourceforge.net/.

Weerawarana Sanjiva [et al.] Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More [Book]. - [s.l.] : Prentice Hall PTR, 2005. - 0-13-148874-0.

WSO2 Lightweight, modular open source middleware for SOA Powered by Apache [Online]. - WSO2. - 9 5, 2008. - http://wso2.com/.