Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Robert Bernhardt

Zur semantischen Geschäftsprozessmodellierung

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Robert Bernhardt

Zur semantischen Geschäftsprozessmodellierung

Bachelorarbeit, eingereicht im Rahmen der Bachelorprüfung
im Studiengang Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Thomas Schmidt
Zweitgutachter: Prof. Dr. rer. nat. Michael Neitzke

Abgegeben am 27. Februar 2008

Thema der Bachelorarbeit

Zur semantischen Geschäftsprozessmodellierung

Stichworte

Geschäftsprozessmodellierung, Semantik, Geschäftsprozess, WSMO, WSMX, WSML, BPMO, BPMN, IP SUPER

Kurzzusammenfassung

Durch die Verwendung von semantischen Geschäftsprozessen ist es möglich den abzubildenen Prozess in einer fachlichen Weise zu modellieren, ohne diesen später in einen technischen Prozess umsetzen zu müssen. Im Vergleich zwischen einem semantischen und technischen Prozess müssen bei einem sematischen Prozess Dienste nicht konkret angegeben werden, sondern nur die gewünschten Zieldefinitionen, welche dazu verwendet werden, den konkreten Dienst erst während der Ausführung zu ermitteln. In der Arbeit werden Technologien und Möglichkeiten aufgeführt, einen semantischen Prozess zu modellieren und auszuführen.

Title of the paper

About the semantic modeling of business processes

Keywords

Semantic, Business Process, WSMO, WSMX, WSML, BPMO, BPMN, IP SUPER

Abstract

By the usage of semantic business processes it is possible to model the process that needs to be illustrated in a specialized way, without the necessity to actual perform the technical process. In the semantic process, services do not need to be defined, but only the final goal, which is used to determine the service during the execution. This thesis will provide technologies and opportunities to execute and model a semantic process.

Inhaltsverzeichnis

1 Einführung

1.1 Motivation

Das ständig wachsende Bedürfnis, Aufgaben in Unternehmen teilweise oder durchgängig zu automatisieren oder auch nur zu unterstützen, führt in gleichem Maße zu einer steigenden Anforderung an IT-Lösungen. Um diesen Anforderungen gerecht zu werden, müssen oft kurzfristige Entscheidungen für Software-Lösungen getroffen werden, wodurch eine ganzheitliche Lösung nicht möglich ist. Die daraus entstandenen gewachsenen IT-Landschaften setzen sich aus einer Vielzahl von einzelnen Komponenten von verschiedensten Anbietern zusammen und besitzten dadurch oft eine unnötig hohe Komplexität. Diese gewachsenen IT-Lösungen führen sehr häufig zu Integrationsproblemen von weiteren Komponenten, da die Kommunikation zwischen diesen Komponenten nicht vollständig standardisiert ist.
Einige Software-Anbieter haben sich in Unternehmensbereichen spezialisiert und können ihren Kunden dadurch eine zugeschnittene IT-Lösung anbieten. Dennoch muss auch diese Lösung an das entsprechene Unternehmen angepasst werden, da sich auch innerhalb einer Branche die Geschäftsprozesse unterscheiden. Trotzdem liegen die damit verbundenen Vorteile auf der Hand. Den Vorteilen für eine solche zugeschnittene Lösung stehen die hohen Kosten und die möglichen Schwierigkeiten bei der Einführung entgegen. Der komplette Umbau einer IT-Landschaft stellt für Unternehmen einen gewaltigen Schritt dar und wird dadurch oft gemieden. Abhilfe für diese Probleme verspricht eine serviceorientierte Architektur (SOA). Unternehmen sehen darin die Möglichkeit, die Probleme der Vergangenheit zu beseitigen ohne einen gesamten Neuanfang mit einem zugeschnittenen Produkt zu wagen und sich auf einen Anbieter festlegen zu müssen. Bei einer SOA handelt es sich in erster Linie um ein Managementkonzept, das eine Ausrichtung der Infrastruktur an die Geschäftsprozesse anstrebt und die Möglichkeit bietet, auf veränderte Anforderungen schnell reagieren zu können. In zweiter Linie setzt eine SOA ein Systemarchitekturkonzept voraus, welches die Bereitstellung von fachlichen Diensten vorsieht. Diese Dienste und Funktionen werden in Form von Services zur Verfügung gestellt und entsprechen atomaren Teilen eines Geschäftsprozesses. Die Kommunikation zwischen den zur Verfügung stehenden Diensten erfolgt auf der Basis einer Standardisierung und ist somit anbieter- und systemunabhängig.
Die Bereitstellung von atomaren Diensten hat den Vorteil, dass der Grad der Wiederverwendbarkeit erhöht und der Implementierungsaufwand minimiert wird, da Funktionen nicht mehrfach implementiert werden müssen. Die Aufteilung von Funktionalitäten in mehrere atomare

Dienste besitzt den Nachteil, diese schwerer verwalten zu können, da die Anzahl der Dienste mit dem Grad der Granularität steigt. Dieser Nachteil kann durch geeignete Software-Komponenten minimiert werden, so dass die Vorteile bei einer Aufteilung der Dienste überwiegen.
Ein Element einer SOA ist ein Service Repository, was die Möglichkeit bietet, die verfügbaren Dienste zu verwalten und damit eine optimalere Übersichtlichkeit zu gewährleisten. Dennoch gibt es bei Umsetzung einer SOA einige Probleme, da Geschäftsprozesse oft von Personen modelliert werden, die einen nicht technischen Hintergrund besitzen. Aus diesem Grund werden Prozesse in einer fachlichen Form modelliert und können dadurch aber nicht direkt von einem System verarbeitet werden. Die Verarbeitung des Prozesses kann erst nach einer Transformation in einen technischen Prozess erfolgen. Dieser Arbeitsschritt wird durch eine Person durchgeführt, die einen technischen Überblick über die Service-Komponenten besitzt bzw. das Know-how besitzt, eine Auswahl der zu verwendenden Dienste zu treffen.

1.2 Zielsetzung

Eine Auswahl von geeigneten Diensten stellt sich in der Praxis, auf Grund ihrer hohen Anzahl, nicht leicht dar. Für ein Element eines fachlichen Geschäftsprozesses muss ein geeigneter Dienst gefunden bzw. eine Kombination aus verschiedenen Diensten gewählt werden, um das gewünschte Ergebnis zu erzielen. Die Suche nach einem entsprechenden Dienst ist eine zeitaufwendige Angelegenheit und wird umso komplexer, je mehr sie über die Unternehmensgrenzen hinausgeht.
Mit steigender Anzahl von Daten sinkt die Wahrscheinlichkeit schnell, relevante Informationen zu finden. Für viele Bereiche wird an einer Lösung des Problems gearbeitet. Ein viel versprechender Ansatz zur Lösung dieses Problems wird mit dem Oberbegriff Semantic Web bezeichnet. Das Konzept sieht vor, dass Inhalte nicht nur für Menschen lesbar sind, sondern auch von Computern interpretiert werden können, wodurch aus dem großen Angebot eine schnelle Vorauswahl getroffen werden kann. Die Anwendung des Konzepts zur Identifizierung passender Dienst würde dazu führen, dass diese Dienste mit zusätzlichen Informationen angereichert werden, die z.B. Auskunft über die Eigenschaften, Verwendung und Qualität des Dienstes geben. Diese Informationen könnten von einem Computer verstanden werden und es könnten Schlüsse gezogen werden, was zu einer enormen Erleichterung bei der Realisierung von Geschäftsprozessen führt. Im einfachsten Fall könnte dieses einem Entwickler die Möglichkeit geben, einen passenden Dienst leichter zu finden, da eine Software die Dienst-Eigenschaften verstehen und nur die relevanten Dienste anzeigen würde. Ein erweitertes Anwendungsbeispiel ist das computergestützte Interpretieren von fachlichen Geschäftsprozessen, bei denen eine Transformation in einen technischen Prozess automatisiert werden kann und im Idealfall kein manueller Eingriff notwendig ist. Ziel dieser Arbeit

ist es, eine Beispielumgebung zu schaffen, in der es möglich ist, fachliche Geschäftsprozesse zu modellieren und diese ohne Transformationsaufwand in eine Systemarchitektur zu integrieren.

2 Semantisches Web

Das Internet stellt eine große Anzahl an Informationen bereit, die von vielen unterschiedlichen Personen erstellt werden. Auf diese Informationen kann auf verschiedenen Wegen zugegriffen werden, indem zum Beispiel direkt eine URI angegeben oder diese über eine Suchmaschine aufgefunden wird. Der bevorzugte Weg auf Informationen zuzugreifen ist Suchen mit einer Suchmaschine, die eine Indizierung von Seiten besitzt und auf der Basis eines Suchalgorithmus ein Ergebnis liefert. Diese Ergebnisliste kann sehr viele Einträge enthalten, die möglicherweise für den Benutzer irrelevant sind, da nur nach Textvorkommen gesucht wird und der Kontext, in dem der Text steht, keinen Einfluss auf das Ergebnis hat.
Die Betreiber von Suchmaschinen entwickeln in diesem Bereich immer bessere Suchalgorithmen, um die Informationsflut zu minimieren und nur relevante Informationen anzuzeigen. Dabei handelt es sich nur um eine syntaktische Volltextsuche, die nicht die Bedeutung der Worte versteht und somit keinen Zusammenhang zwischen dem Sinngehalt der Anfrage und dem möglichen Ergebnis herstellen kann.
Das Ziel des semantischen Web ist es, die Inhalte so zur Verfügung zu stellen, dass auch Maschinen diese interpretieren können und dadurch Zusammenhänge gebildet und Bedeutungen erkannt werden können. Ein Vergleich der Strukturen zwischen dem aktuellen Web und dem semantischen Web ist in der Abbildung 2.1 zu sehen.

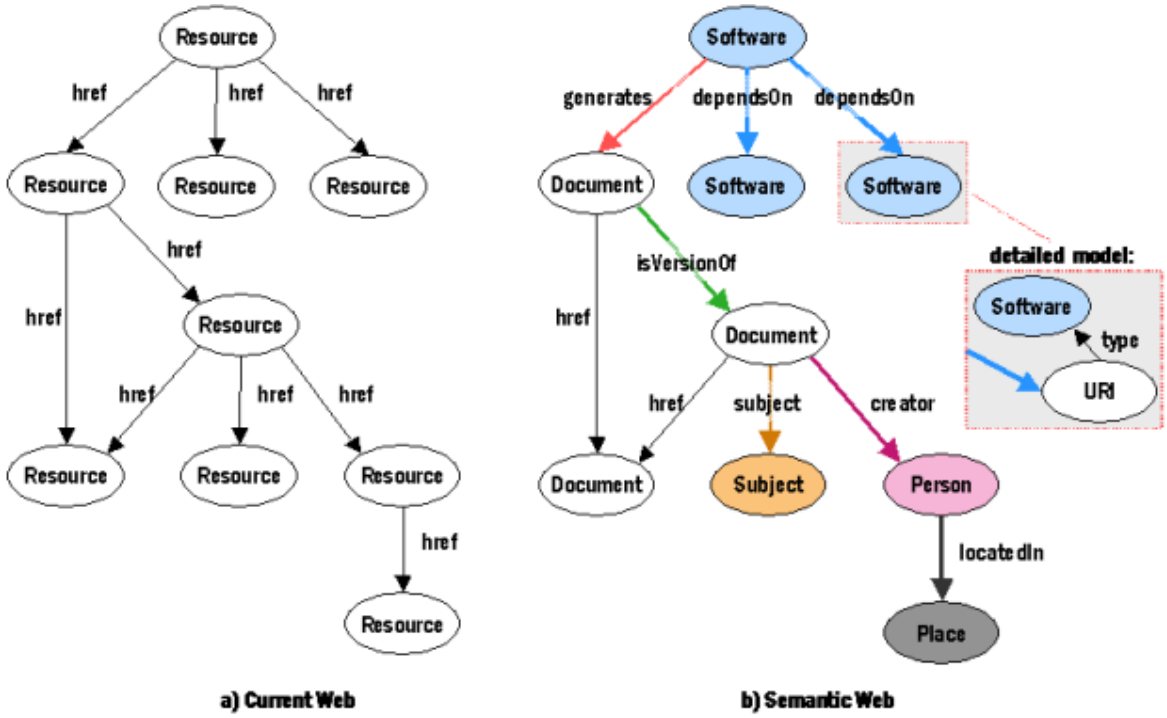

Abbildung 2.1: Vergleich Web – Semantisches Web (Koivunen und Miller, 2001)

Das semantische Web wurde als eine Erweiterung des bestehenden World Wide Web konzipiert und bietet die Möglichkeit, die vorhandenen Informationen mit weiteren maschinen-interpretierbaren anzureichern. Diese Erweiterung wird durch Ontologien ermöglicht, die ein formales System von Begriffen, Relationen und Regeln definieren.

2.1 Ontologien

Der Begriff Ontologie hat seinen Ursprung in der Philosophie und beschreibt die Lehre vom Sein und vom Seienden.

> Ontologie (ontologia): Wissenschaft vom Sein, vom Seienden (on) als solchem, von den allgemeinsten, fundamentalen, konstitutiven Seinsbestimmungen (= allgemeine Metaphysik, s. d.: prôtê philosophia des ARISTOTELES) (Eisler, 1904)

In der Informatik versteht man Ontologien als ein formal definiertes System von Relationen und Begriffen.

> A specification of a representational vocabulary for a shared domain of discourse – definitions of classes, relations, functions, and other objects – is called an ontology. An ontology is an explicit specification of a conceptualization (Gruber, 1993).

Eine klare allgemeine Definition ist in der Informatik für den Begriff der „Ontologie“ nicht ohne Weiteres zu finden, da nicht klar abgegrenzt werden kann, welche Formate und Konzepte genau dazugezählt werden können.

Ontologien wurden im Bereich der künstlichen Intelligenz ab Anfang 1990 populär (Gruber, 1993), wobei ähnliche Ansätze bereits weit früher existierten. Seit dieser Zeit wurde die explizite Formalisierung des Begriffs „Ontologie“ verwendet und fand z.B. in der Forschung im Bereich Künstliche Intelligenz Anwendung. Die Vision des Semantic Web, welche Erwähnung in einer Mitschrift über ein Gespräch mit Tim Berners-Lee fand (Berners-Lee, 1999), rückte das Thema Ontologien wieder in den Vordergrund.

Im Vergleich zu einem reinen Datensatz enthält eine Ontologie zusätzlich formale Beschreibungen und Zusammenhänge von Daten. Daraus lassen sich Rückschlüsse ziehen, Widersprüche erkennen und selbstständig fehlendes Wissen ergänzen. Dieses wird durch Inferenzen (logisches Folgern) ermöglicht.

Eine Grundüberlegung sieht vor, dass Ontologien in Fachbereichen von den jeweiligen Spezialisten modelliert werden, diese dann zur Verfügung gestellt und weiterverwendet werden können. Diese Überlegung stellt sich in der Praxis als schwer realisierbar dar, da zwischen den Experten des Bereichs eine Einigkeit herrschen muss. Weiterhin ist Wissen von einer Perspektive abhängig und es besteht ein unterschiedliches Bedürfnis an Informationen zu einem Bereich. Aus diesem Grund bietet es sich an, Ontologien in einer Taxonomie zu strukturieren.

- Allgemeine Ontologien – Diese Art von Ontologien stellt allgemeines Wissen zur Verfügung (z.B. eine Ontologie der Zeit[1]).
- Ontologien in Bereichen – Ontologien in dieser Ebene können Wissen zu speziellen Bereichen oder Unternehmen zur Verfügung stellen z.B. eine ontologiebasierte Warenübersicht eines Versandunternehmens.
- Spezielle Ontologien – Eine spezielle Ontologie stellt das Wissen für einen Anwendungsfall bereit. In Bezug auf das Beispiel des Versandunternehmens könnte eine Ontologie die Eigenschaften einer Produktsuche repräsentieren.

Eine exakte Zuordnung einer Ontologie zu einer Ebene ist in der Praxis nicht immer möglich.

[1] http://www.wsmo.org/ontologies/dateTime/

2.2 Technische Komponenten

Eine wichtige Komponente des semantischen Web sind Ontologien, dennoch gibt es erheblich mehr Elemente. Diese Elemente bieten erst im Zusammenspiel die gewünschte Funktionalität des semantischen Web. In der Abbildung 2.2 ist das aktuelle Schichtenmodell dargestellt, wobei dieses noch nicht als endgültige Lösung anzusehen ist.

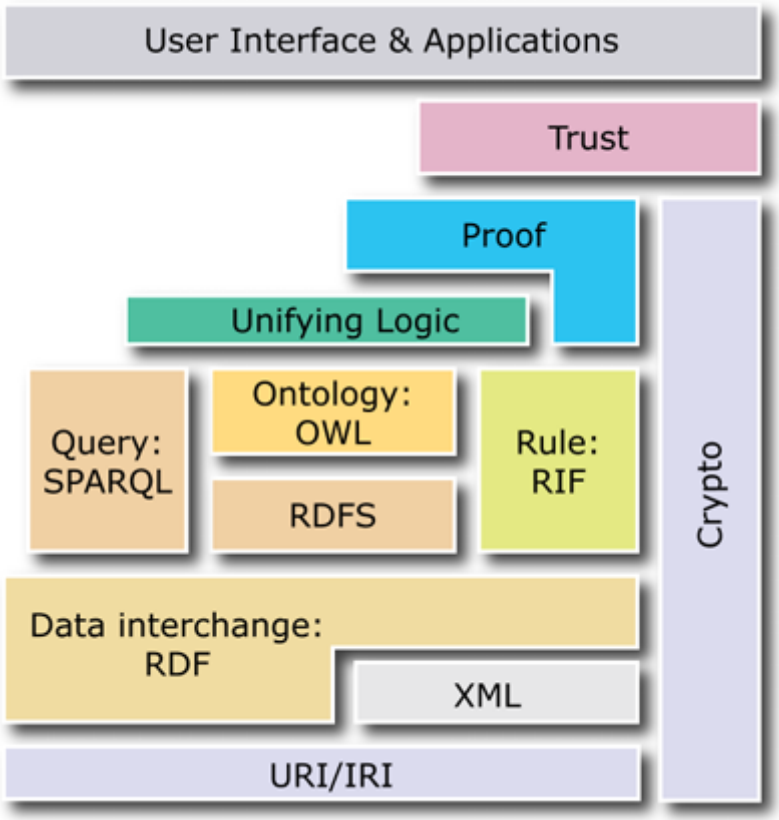

Abbildung 2.2: Semantic Web Schichtenmodell (Berners-Lee u. a., 2007)

Es herrscht im Moment immer noch Uneinigkeit, ob die dargestellten Komponenten korrekt angeordnet sind und ob damit ein ideales Zusammenspiel ermöglicht wird. Die in der letzten Zeit in kurzen Abständen veröffentlichten Versionen des Schichtenmodells sorgen jeweils für Diskussionen und es ist nicht abzusehen, wann vorerst eine finale Version vorliegt. Ein Grund für die Uneinigkeit sind die meist noch nicht vorhandenen Technologie-Empfehlungen. Dennoch ist im Verhältnis zum originalen[2] Schichtenmodell eine erhebliche Entwicklung zu erkennen, da einige Elemente durch Empfehlungen des W3C spezifiziert wurden. In diesem Abschnitt wird nur auf einige dieser Komponenten eingegangen, um eine Grundlage für die

[2] http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html

weiteren Kapitel zu schaffen. Weiterführende Informationen sind z.B. auf der Internetpräsenz der Arbeitsgrupppe „Semantic Web“ zu finden (Berners-Lee u. a., 2007).

2.2.1 URI/IRI

Die untere Ebene des Schichtenmodells stellt die Möglichkeit dar, Ressourcen anzusprechen und eindeutig zu identifizieren. Dies wird durch Uniform Resource Identifier (URI[3]) und Internationalized Resource Identifier (IRI[4]) ermöglicht. Beide Technologien wurden durch die IETF standardisiert und werden in allen Bereichen des Internets eingesetzt. IRIs sind eine Ergänzung von URIs und erweitern diese um den Unicode Zeichensatz (Unicode / ISO 10646). Dabei wird die Kompatibilität zum weit verbreiteten Uniform Resource Identifier gewährleistet, indem eine IRI auch als URI dargestellt werden kann.

Beispiele:

- URI – `http://resume.example.org`
- IRI – `http://résumé.example.org`
- IRI als URI – `http://xn--rsum-bpad.example.org`

2.2.2 XML und XML Schemas

Die Extensible Markup Language (XML) ist eine Meta-Sprache, die einer Teilmenge von SGML[5] entspricht. Im Vergleich zu SGML zeichnet sich die Extensible Markup Language durch ihre Einfachheit, aber einen dennoch weiterhin vorhandenen hohen Grad an Mächtigkeit aus.
Durch XML ist es möglich, Daten hierarchisch strukturiert darzustellen, wobei durch die Spezifikation[6] von XML keine Vorgaben für die Strukturierung gemacht werden. Diese Eigenschaft hat den Vorteil, dass XML in vielen Bereichen eingesetzt werden kann. Dennoch ist es möglich, die Struktur eines XML-Dokuments durch ein Schema (XSD) zu definieren und dadurch unter anderem für eine syntaktische Kompatibilität zwischen Systemen zu sorgen.
Damit ein XML-Dokument verarbeitet werden kann, muss es eine oder, wenn ein Schema angegeben wurde, zwei Eigenschaften besitzen. In erster Line muss ein Dokument die Spezifikation erfüllen, um als wohlgeformt bezeichnet werden zu können. Ein wohlgeformtes Dokument muss unter anderem genau ein Wurzelelement besitzen und die Bennennung eines

[3] `http://tools.ietf.org/html/rfc1630`
[4] `http://tools.ietf.org/html/rfc3987`
[5] Standard Generalized Markup Language (ISO 8879)
[6] `http://www.w3.org/TR/xml/`

Attributes muss innerhalb eines Elements (Tag) eindeutig sein. Sobald ein Schema angegeben wurde, müssen auch diese Vorgaben erfüllt werden und erst dann kann das Dokument als gültig bezeichnet werden.
Das nachfolgende Beispiel zeigt ein XML-Dokument, dessen Struktur durch ein Schema definiert wurde und nur dann eine Gültigkeit besitzt, wenn die Vorgaben des Schemas erfüllt werden.

Listing 2.1: XML-Dokument

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<bachelorarbeit>
    <titel>Zur semantischen Geschäftsprozessmodellierung</titel>
    <author>
        <name>Robert Bernhardt</name>
        <mail>bernhardt@newsarea.de</mail>
    </author>
</bachelorarbeit>
```

Das XSD-Beispiel zeigt die Definition für den Aufbau des XML-Dokuments.

Listing 2.2: XSD-Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="
   qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="bachelorarbeit">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="titel" type="xs:string" />
        <xs:element name="author">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="name" type="xs:string" />
              <xs:element name="mail" type="xs:string" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Die Verarbeitung von XML- Dokumenten kann, unabhängig von Strukturierung und Inhalt, durch einen Standardprozessor durchgeführt werden. Diese Eigenschaft stellt einen weiteren großen Vorteil von XML dar.

2.2.3 RDF

Durch XML ist es möglich, syntaktische Strukturen zu definieren und Daten darzustellen, aber diese können nicht semantisch beschrieben werden. Eine semantische Beschreibung muss extern spezifiziert werden, wofür das Resource Description Framework verwendet werden kann.
Mit Hilfe des Resource Description Framework (RDF) ist es möglich, die Daten mit Bedeutungen zu versehen, so dass eine Maschine deren Bedeutung erkennt und entsprechend verarbeiten kann. RDF ist eine formale Sprache zur Anreicherung von Inhalten mit Meta-Daten. RDF basiert auf der Idee, dass Dinge Eigenschaften besitzen, denen Werte zugeordnet sind, und man ein Element durch Angaben beschreiben kann (Manola und Miller, 2004). Die Realisierung dieser Idee wird durch Statements ermöglicht, welche aus drei Angaben bestehen (Subjekt, Prädikat und Objekt). Durch die Angabe dieser Informationen kann ein Element beschrieben werden und es können diese Informationen sowohl von einem Menschen als auch von einer Maschine gelesen und interpretiert werden.

Listing 2.3: RDF-Statement (Manola und Miller, 2004)

```
http://www.example.org/index.html  has a creator whose value is
   John Smith
```

Im Beispiel wird eine Internet Ressource mit der Information des Erstellers versehen. Das angegebene Statement kann wie folgt den Elementen des RDF-Triple zugeordnet werden.

Subjekt http://www.example.org/index.html

Prädikat creator

Objekt John Smith

Die Abbildung 2.3 zeigt den zugehörigen RDF Graphen.

Auf weitere Informationen wird an dieser Stelle verzichtet, da nur ein sehr grober Überblick vermittelt werden soll. Weitere Informationen zu RDF sind in der Spezifikation zu RDF zu finden (Manola und Miller, 2004).

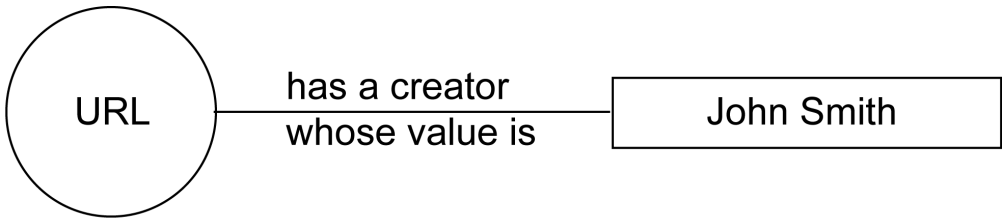

Abbildung 2.3: RDF Graph

2.2.4 OWL

Eine erweiterte Sprache zur Beschreibung von Ontologien ist die Web Ontologie Language (OWL). OWL wurde von DAML+OIL abgeleitet und basiert auf RDF + RDFS. Im Vergleich zu RDF + RDFS besitzt OWL ein standardisiertes Schema[7] zur Modellierung von Ontologien und erweitert RDF + RDFS z.B. um eine Aussagenlogik und eine erweiterte Aussagefähigkeit. Die Aussagefähigkeiten einer Ontologie lassen sich bei OWL in drei Teile unterteilen. Je nachdem welche Aussagefähigkeit benötigt wird, kann eine dieser OWL-Arten verwendet werden. Eine wichtige Unterscheidung zwischen OWL Full und den anderen beiden Sprachebenen ist die bei OWL Full nicht gewährleistete Prozessierbarkeit.

- **OWL Full** enthält das gesamte Sprachspektrum von OWL und stellt somit die Gesamtheit der Leistungsmerkmale dar. Die Nutzung von OWL Full und die nicht vorhandenen Einschränkungen haben den Nachteil, dass die Prozessierbarkeit durch einen Reasoner nicht gewährleistet werden kann. Der Grund dafür liegt z.B. in der Möglichkeit, dass Klassen andere Klassen instanzieren können.
- **OWL DL** besitzt im Gegensatz zu OWL Full einige Einschränkungen, wodurch eine Prozessierbarkeit der Ontologie gewährleistet werden kann. Die Eigenschaft einer Ontologie, endlich prozessierbar zu sein, fordert die Einschränkungen, dass nur Prädikationslogiken der ersten Stufe ausgedrückt werden können und eine endliche Aufzählbarkeit von Mengen gewährleistet sein muss.
- **OWL Lite** kann zur Erstellung von einfachen Ontologien verwendet werden und enthält nur Basiskonstrukte. Dieses hat den Vorteil, dass durch die geringe Komplexität schneller ein Ergebnis erziehlt werden kann.

OWL lässt sich grob in drei Elemente unterteilen, welche durch die Konstrukte der Spezifikation (Bechhofer u. a., 2004) beschrieben werden können. Eine mit OWL erstellte Ontologie kann aus einem Modell, Daten und Regeln bestehen. Das Modell lässt sich durch Klassen

[7] http://www.w3.org/2002/07/owl

und Eigenschaften beschreiben und die Daten werden durch Instanzen, welche als Basis eine oder mehrere Klassen besitzen können, angegeben. Die Regeln einer Ontologie können mit Axiomen definiert werden, und eine mögliche Verwendung ist die Sicherstellung von korrekten Daten.

3 Semantische Web Services

Ein spezielles Gebiet des semantischen Web sind semantische Web Services. Genau wie bei den allgemeinen Ressourcen des Web ist es gewünscht, die Daten und Funktionalitäten mit Bedeutung zu versehen und diese von Maschinen interpretierbar zu machen. Dies wird durch die semantische Beschreibung erreicht.
Web Services stellen ihre Funktionalitäten in Form einer standardisierten Schnittstelle (Abschnitt 3.1.1) zur Verfügung, welche von Softwarekomponenten verwendet werden kann. Diese auf IT-Systeme ausgerichtete Schnittstelle enthält alle Informationen, um die Funktionen des Dienstes nutzen zu können. Eine Einbindung in eine Software erfolgt durch Angabe der technischen Beschreibung des Dienstes und gibt Auskunft über die zur Verfügung stehenden Methoden und deren Ein- und Ausgabeparameter. Eine Aussage über die Bedeutungen oder Funktionalitäten wird nicht getroffen.
Erst durch eine semantische Beschreibung der Schnittstelle wird einer Maschine die Möglichkeit gegeben, die Funktionalität des Dienstes zu interpretieren. Die Aussicht, dass ein System das Leistungsspektrum von Web Services erkennen kann, bietet eine Vielzahl von Anwendungsmöglichkeiten und Erleichterung, welche in den folgenden Kapiteln gezeigt werden.
Durch das große Interesse und die intensiven Bemühungen in diesem Bereich gibt es verschiedene Technologien, an denen derzeit gearbeitet wird. Es ist im Moment nicht abzusehen, welche dieser Techniken sich letztendlich durchsetzen wird, aber dennoch ist bei einigen bereits ein großes Spektrum an Funktionalitäten realisiert. Generell wird bei allen Entwicklungen der gleiche Ansatz verfolgt.
Eine der am weitesten entwickelten Technologien ist die Web Service Modelling Ontology (WSMO), bei der es sich um ein Metamodell handelt. Ein Grund für die weite Entwicklung von WSMO ist sicherlich die Zugehörigkeit der WSMO Arbeitsgrupppe zur European Semantic Systems Initiative (ESSI[1]). Die European Semantic Systems Initiative treibt die Entwicklung voran, da weitere Projekte der ESSI auf den Konzepten von WSMO basieren. Das ESSI SUPER Projekt basiert z.B. in großen Teilen auf WSMO und dem zugehörigen Konsortium sind einige namenhafte Unternehmen angeschlossen.

[1] http://www.essi-cluster.org

3.1 Web Services

Die Basisfunktionalitäten von Web Services werden durch zwei Haupt-Komponenten, oberhalb der Anwendungsprotokoll-Ebene (HTTP, FTP etc.), zur Verfügung gestellt. Die technische Beschreibung für einen Aufruf der Web Service Methoden erfolgt durch eine Schnittstellen-Spezifikation (WSDL), auf deren Basis eine Ausführung einer entfernten Methode durchgeführt werden kann, was mit Hilfe des SOAP Protokolls realisiert wird. Ein sinnvolle, aber nicht notwendige Komponente im Umgang mit Web Services ist ein Service-Repository. In diesem können die verschiedenen Services veröffentlicht, verwaltet und gefunden werden. Im Gegensatz zu den technischen Basis-Komponenten hat sich für ein Service-Repository noch keine Standardisierung durchsetzen können. Im Allgemeinen gibt es zwei unterschiedliche Ansätze. Einer verfolgt den Ansatz der zentralen (z.B. UDDI) und der andere der dezentralen (z.B. WSIL) Bereitstellung der Services.

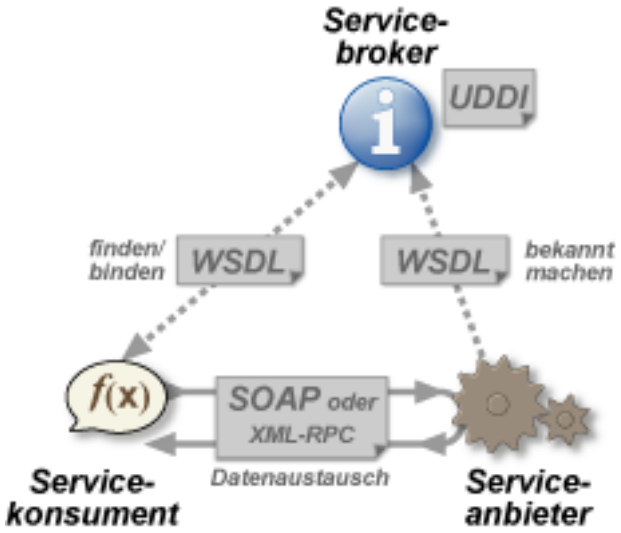

Abbildung 3.1: Funktionsweise von Web Services (Vömchen, 2005)

3.1.1 WSDL

Bei der Web Services Description Language (WSDL) handelt es sich um eine XML-basierte Sprache zur Beschreibung von technischen Funktionen eines Web Service. In der Beschreibung eines Dienstes sind folgende Grundinformationen enthalten (Christensen u. a., 2001).

- **Types** – Datentypen die zum Austausch der Nachrichten verwendet werden
- **Message** – abstrakte Definition der übertragenen Nachricht und Daten

- **Operation** – Methoden, die vom Dienst zur Verfügung gestellt werden
- **Port Type** – abstrakte Menge von Operationen, welche einen oder mehrere Endpunkte unterstützen können
- **Binding** – konkrete Bindung eines Port Types an ein Protokoll und ein Datenformat
- **Port** – Kommunikationsschnittstelle als Internet Ressource
- **Service** – Menge von ähnlichen Ports

Die beschriebenen Daten liefern alle Informationen, um einen Aufruf einer entfernten Methode durchführen zu können (Remote Method Invocation).

3.1.2 SOAP

SOAP ist ein Netzwerkprotokoll für den Austausch von strukturierten Informationen. Im Umgang mit Web Services wird SOAP verwendet, um eine Ausführung einer entfernten Methode zu ermöglichen. Die Struktur der Nachricht und der Endpunkt, an den die Nachricht versendet wird, ist durch die Schnittstellenbeschreibung (WSDL) vorgegeben. Die Kommunikation mit dem angegebenen Web Service Endpunkt erfolgt meistens über das HTTP und TCP Protokoll, obwohl keine Beschränkungen für das Transportprotokoll vorliegen.
Der Grundaufbau eine SOAP Nachricht ist in der Abbildung 3.2[2] zu sehen. Die Hülle wird als „Envelope“ bezeichnet und enthält die Elemente „Header“ und „Body“.

3.2 WSMO

Im vorigen Abschnitt wurde ein kleiner Überblick über die Funktionen und Funktionsweisen von Web Services gegeben. Diese Informationen stellen die Grundlage für die Verwendung eines Web Service dar, welche nun um die Möglichkeit erweitert werden, eine semantische Beschreibung durchführen zu können. Für diesen Zweck wird die Web Service Modelling Ontology verwendet. Bei der WSMO handelt es sich zum einen um ein konzeptionelles Modell zur Beschreibung von semantischen Diensten und zum anderen um eine Ontologie für diesen Zweck.
Die Entwicklung von WSMO gliedert sich in drei Teile (WSMO, WSML und WSMX), die in Arbeitsgruppen aufgeteilt sind. Diese Arbeitsgruppen sind Teil der European Semantic Systems Initiative (ESSI[3]).

[2] Quelle: http://www.sei.cmu.edu/isis/guide/technologies/web-services.htm
[3] http://www.essi-cluster.org/

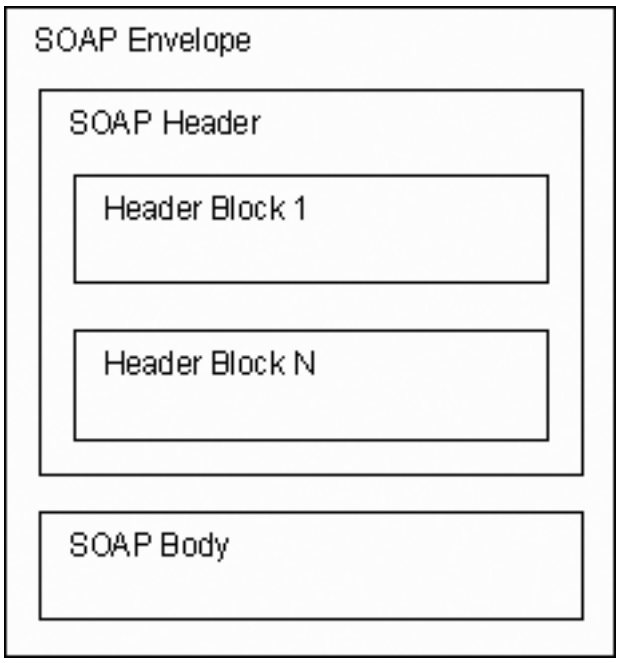

Abbildung 3.2: Aufbau einer SOAP Nachricht

Web Service Modelling Ontology besitzt vier Hauptelemente (siehe Abbildung 3.3), die alle benötigten Funktionalitäten bereitstellen, um Dienste semantisch zu beschreiben.

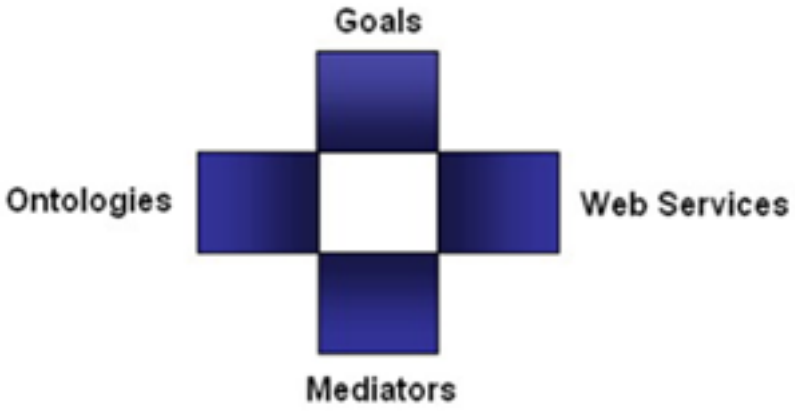

Abbildung 3.3: WSMO Elemente (de Bruijn u. a., 2005)

3.2.1 Ontologien (Ontologies)

Ontologien wurden bereits im Abschnitt 2.1 beschrieben. Die Verwendung von Ontologien ist im Modellkonzept von WSMO ähnlich der von OWL (Abschnitt 2.2.4). Dennoch gibt

es Unterschiede bei den möglichen Eigenschaften einer Ontologie. In den Spezifikationen von WSMO (de Bruijn u. a., 2005) werden diese möglichen Eigenschaften als Ontologie-Definition (Listing 3.1) aufgeführt. Der größte Unterschied und auch Vorteil ist die Möglichkeit, Vermittler (Mediator) verwenden zu können und somit Ontologien untereinander zu verknüpfen. Die Eigenschaft „importsOntology“ verknüpft zwar auch Ontologien untereinander, genau wie bei der Verwendung von Vermittlern, bietet aber nur die Möglichkeit, Elemente aus anderen Ontologien zu verwenden und dient nicht zur Auflösung von Kompatibilitätsproblemen.
Die Umgang mit Mediatoren wird in Passage 3.2.4 näher erläutert.

Listing 3.1: Ontologie-Definition

```
Class ontology
    hasNonFunctionalProperties type nonFunctionalProperties
    importsOntology type ontology
    usesMediator type ooMediator
    hasConcept type concept
    hasRelation type relation
    hasFunction type function hasInstance type
    instance hasAxiom type axiom
```

Ein weiterer Eintrag in den Definitionen sind die nicht-funktionalen Eigenschaften (Listing 3.2) einer Ontologie, welche dazu verwendet werden können, z.B. Informationen zum Ersteller zu hinterlegen. Die Möglichkeit, nicht-funktionale Eigenschaften zu hinterlegen, steht für jedes WSMO-Element zur Verfügung.

Listing 3.2: Nicht-funktionale Eigenschaften

```
Class nonFunctionalProperties
    hasAccuracy type accuracy
    hasContributor type dc:contributor
    hasCoverage type dc:coverage
    hasCreator type dc:creator
    hasDate type dc:date
    hasDescription type dc:description
    hasFinancial type financial
    hasFormat type dc:format
    hasIdentifier type dc:identifier
    hasLanguage type dc:language
    hasNetworkRelatedQoS type networkRelatedQoS
    hasOwner type owner
    hasPerformance type performance
    hasPublisher type dc:publisher
```

```
16      hasRelation type dc:relation
        hasReliability type reliability
18      hasRights type dc:rights
        hasRobustness type robustness
20      hasScalability type scalability
        hasSecurity type security
22      hasSource type dc:source
        hasSubject type dc:subject
24      hasTitle type dc:title
        hasTransactional type transactional
26      hasTrust type trust
        hasType type dc:type
28      hasTypeOfMatch type typeOfMatch
        hasVersion type version
```

In den nachfolgenden Abschnitten wird auf die noch nicht erläuterten Elemente der Ontologie Definition (Listing 3.1) eingegangen.

Konzepte (Concepts)

Konzepte sind Teile einer Ontologie und beschreiben ein Modell, was mit Klassen in der objektorientierten Programmierung verglichen werden kann. Die möglichen Eigenschaften eines Elements können angegeben werden, ohne diesen wirkliche Werte zuzuweisen. Im Beispiel 3.3 wird z.B. ein Konzept für eine Telefonnummer dargestellt. Durch eine Instanz könnte aus diesem Modell eine Repräsentation einer Telefonnummer erstellt werden und die definierten Eigenschaften mit Werten versehen werden. Der Vergleich zur OOP wäre die Erstellung eines Exemplares einer Klasse und der Zuweisung von Werten.

Listing 3.3: Ausschnitt der Communication Ontologie

```
1  concept ContactInfo

3  concept PhoneNumber subConceptOf ContactInfo
       countryCode impliesType _string
5      prefix impliesType _string
       number impliesType _string
7      extension impliesType _string
       network impliesType PhoneNetwork
9
   concept PhoneNetwork
11     name impliesType _string
```

```
    country impliesType loc#Country

concept CellularPhoneNetwork subConceptOf PhoneNetwork
    prefix impliesType _string
```

Eine weitere Eigenschaft von Konzepten ist die Mehrfachvererbung dieser. Im Beispiel 3.3 wird das Konzept PhoneNumber von ContactInfo abgeleitet. Bei einer Vererbung erhält das SubConcept alle Eigenschaften, von dem es abgeleitet wurde. ContactInfo besitzt im Beispiel keine Eigenschaften und es werden somit auch keine vererbt. Die Ableitung wurde gewählt, um ein mögliches weiteres ContactInfo Konzept (z.B. EMailAddress) erstellen zu können und diese durch die Ableitung mit PhoneNumber zu gruppieren.
Durch die Mehrfachvererbung ist es möglich, auch weitere Konzepte anzugeben, von denen das SubConcept jeweils alle Eigenschaften erhalten würde.

Eigenschaften von Konzepten können einen Basistyp darstellen oder auf andere Konzepte verweisen. Der Wert einer Eigenschaft mit dem Typ eines Konzepts ist eine Instanz dieses.

Instanzen (Instances)

Eine Instanz ist ein Exemplar eines Konzepts und gibt für die Eigenschaften des Konzepts die Werte an. Das Beispiel 3.4 zeigt einen Telefonanbieter mit den Eigenschaften des „CellularPhoneNetwork“ Konzepts.

Listing 3.4: O2 Germany Instanz von CellularPhoneNetwork

```
instance O2_Germany memberOf CellularPhoneNetwork
    prefix hasValue {"0179", "0176" }
    name hasValue "O2 (Germany) GmbH & Co. OHG"
    country hasValue loc#Germany
```

Axiome (Axioms)

In den vorigen beiden Teilbereichen wurden Konzepte und Instanzen beschrieben. Die Werte einer Instanz sind nur in Hinsicht auf den Typ der Eigenschaft beschränkt, was zu Fehleingaben führen könnte. Eine Möglichkeit diese Fehleingaben zu vermeiden oder aus Eigenschaften andere abzuleiten, ist die Verwendung von Axiomen. Ein Axiom kann als Regel verstanden werden und definiert Einschränkungen oder Zusammenhänge. In dem Beispiel 3.3 kann zum Beispiel eine Eingabe fehlerhaft sein, wenn eine Instanz von „PhoneNumber“ gebildet wird, der Wert von CountryCode „0049“ entspricht und die Telefonnummer einem

Netzwerk aus Großbritannien zugeordnet wird. Diese Fehleingabe könnte durch eine Einschränkung vermieden werden. Ein weitere Möglichkeit für die Verwendung von Axiomen ist die Erweiterung von Eigenschaften. Das gebildete Axiom 3.5 erweitert z.B. die Eigenschaft eines „PhoneNumber“-Konzepts automatisch um das Netzwerk, sobald eine Vorwahl angegeben wird.
In Ontologien können durch Axiome logische Ausdrücke erstellt werden, wodurch die Korrektheit der Daten sichergestellt werden kann und die Möglichkeit besteht, implizit Instanzen von Konzepten zu erweitern.

Listing 3.5: Axiom zur Identifizierung eines Telefonanbieters

```
axiom CellularPhoneNetworkByPrefix
    definedBy
        ?phNum[
            prefix hasValue ?prefix
        ] memberOf PhoneNumber
        and ?network[
            prefix hasValue ?prefix
        ] memberOf CellularPhoneNetwork
        implies ?phNum[
            network hasValue ?network
        ].
```

Beziehungen (Relations)

Beziehungen werden verwendet, um eine Abhängigkeit zwischen Konzepten oder deren Instanzen zu erzeugen.

Listing 3.6: Definition einer Relation

```
entity relation
    nonFunctionalProperties ofType nonFunctionalProperties
    superRelations ofTypeSet relation
    parameters ofTypeSet parameter
    definedBy ofType logicalExpression
```

Funktionen (Functions)

Bei einer Funktion handelt es sich um eine spezielle Art von Relation, die einen Rückgabe-Wert besitzt. Durch Funktionen können z.B. mathematische Berechnungen durchgeführt werden.

Listing 3.7: Definition einer Funktion

```
entity function subEntityOf relation
    range ofType concept
```

3.2.2 Web Services

Web Services werden mit WSMO nur semantisch beschrieben, wobei diese Beschreibung mehrere Komponenten besitzt. Die Elemente, die zur Beschreibung verwendet werden können, sind die Fähigkeit (Capability), die Schnittstellen, verknüpfte Ontologien und mögliche benötigte Vermittler (Mediators). Die Informationen über die Fähigkeit geben Auskunft über das Leistungsvermögen des Service.

```
Class capability
    hasNonFunctionalProperties type nonFunctionalProperties
    importsOntology type ontology
    usesMediator type {ooMediator, wgMediator}
    hasSharedVariables type sharedVariables
    hasPrecondition type axiom
    hasAssumption type axiom
    hasPostcondition type axiom
    hasEffect type axiom
```

Diese Funktionalität kann unter anderem durch Beschreibungen bei vier Ereignissen eines Aufrufs definiert werden.

- Die Vorbedingung (**Precondition**) gibt Auskuft über den Zustand vor einer Nutzung des Dienstes und dient z.B. zur Beschreibung der notwendigen Informationen für einen Aufruf.
- Eine Voraussetzung (**Assumption**) stellt die Situation der Umgebung dar, die vor der Ausführung benötigt wird.
- Nachbedingungen (**Postcondition**) beschreiben das Result der Ausführung in Bezug auf den aktuellen Kontext und die Eingabewerte des Aufrufs.
- Die Auswirkung auf die Umgebung kann an letzter Stelle der Ausführung (**Effect**) beschrieben werden, wobei diese sich nicht zwangsläufig auf die Eingaben oder den aktuellen Kontext beziehen muss.

Der nachfolgende Ausschnitt aus einer Web Service Beschreibung zeigt ein Beispiel für eine Nachbedingung (Postcondition). Nach der Definition besitzt der Service die Möglichkeit Versendungen durchzuführen, solange es sich bei dem Empfänger um ein Telefon handelt,

welches vom O2 Germany Anbieter zur Verfügung gestellt wird.
Die verwendete Ontologie mit dem Prefix „co“ ist im Anhang zu finden 8.3.

Listing 3.8: PostCondition – O2SMSService Web Service

```
postcondition
    definedBy
        ?sms[
            co#receiver hasValue ?receiver
        ] memberOf co#Transmission
        and ?receiver[
            co#network hasValue co#O2_Germany
        ] memberOf co#PhoneNumber.
```

Eine weitere Komponente der Beschreibung eines Web Service, ist die Angabe von Schnittstellen. Schnittstellen definieren, wie eine Aktion ausgeführt werden kann, um die gewünschte Funkionalität zu erhalten. Bei dieser Beschreibung wird zwischen zwei Arten unterschieden.

- **Choreography** beschreibt die Kommunikation mit einem Web Service vom Standpunkt des Teilnehmers.
- **Orchestration** wird verwendet, um einem Teilnehmer einen Dienst zur Verfügung stellen zu können, ohne dass dieser von seinem Standpunkt erkennen kann, welche weiteren Dienste einbezogen werden, um die Fähigkeiten sicherzustellen.

Ein weiterer Auszug aus dem SMS Service zeigt die Schnittstelle mit der notwendigen Choreografie. Die Eingabe und Ausgabeparamter sind durch die Ontologie-Konzepte Transmission und Confirmation beschrieben. Für die Ausführung des Web Service wird die WSDL-Beschreibung und die auszuführende Methode benötigt, welche im Zusammenhang mit „withGrounding“ angegeben wird.

Listing 3.9: Interface – O2SMSService Web Service

```
interface SendInterface
  choreography chor
    stateSignature sign
      importsOntology {
        _"http://www.newsarea.de/wsmo/ontologies/newsarea/v1.0/
            Newsarea.wsml",
        _"http://www.newsarea.de/wsmo/ontologies/communication/v1
            .0/Communication.wsml"
      }

```

```
    in
      concept co#Transmission withGrounding _"http://services.
          rappcollins.de/temp/o2sms/1.0/SMSService.asmx?WSDL#wsdl.
          interfaceMessageReference(SMSServiceSoap/Send/In)"
    out
      concept na#Confirmation
```

3.2.3 Ziele (Goals)

WSMO verfolgt das Konzept des Goal-Driven Approach, welches die Idee besitzt, eine Zieldefinition zu modellieren. Ein Goal entspricht dem Gegenstück einer Web Service Beschreibung.
Aus der Sicht des Service-Konsumenten werden die Funktionalitäten (Capability) angegegeben, die zur Erfüllung des Ziels benötigt werden. Dieses definierte Ziel kann durch eine Discovery Engine, welche mit einem Service Repository verbunden ist, die benötigten Komponenten herausfinden.
Durch die Ausführung des SendO2SMS (Listing 3.10) Goals würde das im Abschnitt „Web Services“ beschriebene Service-Beispiel gefunden werden. Bei einem Vergleich der Nachbedingungen (Postcondition) ist zu erkennen, dass ein Goal nicht exakt die gleiche Beschreibung verwenden muss, sondern sich die Funktionalität nur aus der Ontologie ableiten lassen muss. Im Beispiel wird das unter anderem durch ein Axiom ermöglicht, was einen Zusammenhang zwischen dem Telefon-Netzwerk und der Vorwahl herstellt (siehe Ontologie 8.3).

Listing 3.10: Capability – SendO2SMS Goal

```
goal _"http://www.newsarea.de/wsmo/goals/SendSMS.wsml"

    importsOntology {
            _"http://www.newsarea.de/wsmo/ontologies/global/v1
                .0/Global.wsml"
    }

capability SendO2SMS

postcondition
  definedBy
    ?sms[
        co#receiver hasValue ?receiver
    ] memberOf co#Transmission
    and ?receiver[
```

```
            co#prefix hasValue "0179"
16        ] memberOf co#PhoneNumber.
```

3.2.4 Vermittler (Mediators)

Für die Zuordnung zwischen WSMO-Komponenten steht die Möglichkeit zur Verfügung Vermittler (Mediators) zu erstellen. Diese Funktion spielt im Bereich von verteilten Systemen eine wichtige Rolle, da an unterschiedlichen Orten und von unterschiedlichen Personen Komponenten modelliert werden, die erst durch eine Verknüpfung miteinander verwendet werden können. Für diese Aufgabe stehen vier Typen zur Verfügung.

- OO Mediators – Ontologien mit Ontologien
- GG Mediators – Ziele mit Zielen
- WW Mediators – Web Services mit Web Services
- WG Mediators – Web Services mit Zielen

Durch Kombination dieser Typen können alle Elemente miteinander verknüpft werden (siehe Abbildung 3.4).

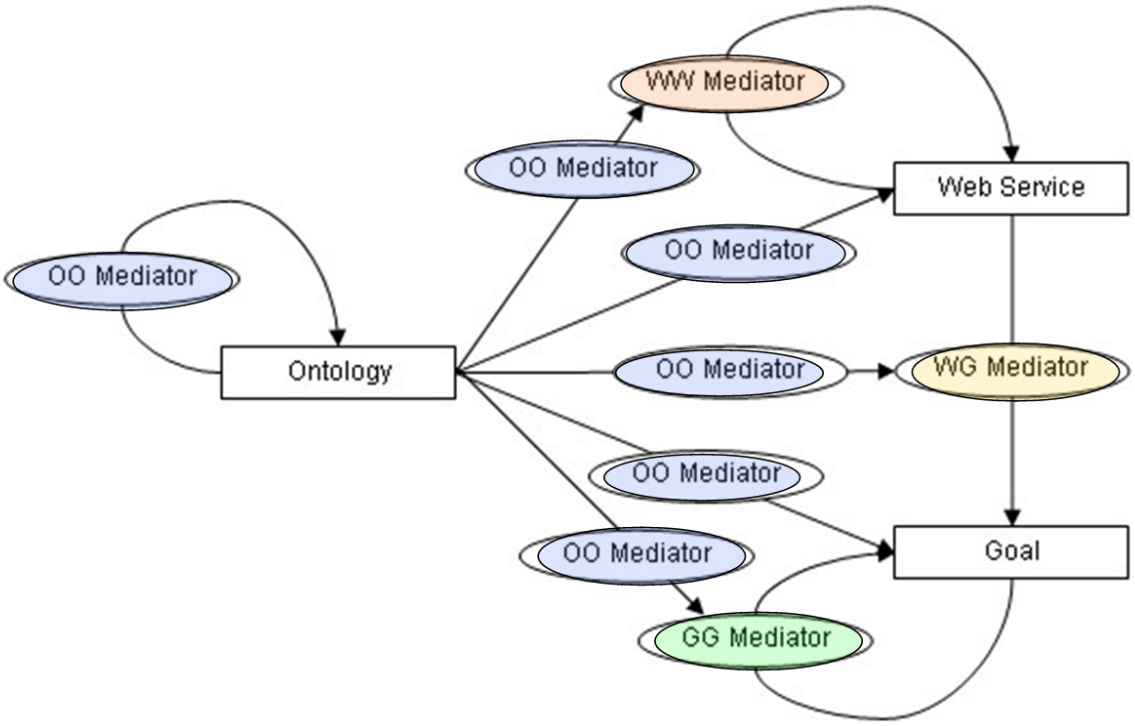

Abbildung 3.4: Kombinationen von Vermittlern (Mediators) (Bussler u. a., 2005)

3.2.5 WSML

Durch die Web Service Modelling Language (WSML) wird eine formale Syntax und Semantik zur Umsetzung der konzeptionellen Elemente von WSMO zur Verfügung gestellt. Die Entwicklung von WSML basiert auf Beschreibungslogiken, Prädikatenlogik der ersten Stufe, Programmier Logik und Frame Logik (Bussler u. a., 2005).
Die Eigenschaft von WSML, in einem menschlich lesbaren Format und einer austauschbaren Syntax erstellt zu werden, sind ein wichtiges Merkmal von WSML.
Bei der Verwendung von WSML kann auf fünf Varianten zurückgegriffen werden.
WSML-Core besitzt von allen Varianten die geringste Ausdruckskraft und kann durch WSML-DL um eine Beschreibungslogik und durch WSML-Flight und WSML-Rule um die Eigenschaft der logischen Programmierung erweitert werden. WSML-Full stellt das gesamte Leistungsspektrum zur Verfügung.
Weitere Informationen zum Leistungsumfang von WSML sind in der Ausarbeitung (Bussler u. a., 2005) zum Anlass der zweiten IEEE International Conference on Autonomic Computing (ICAC-05) zu finden.

3.2.6 WSMX

Die Implementierung des WSMO-Konzepts stellt die Web Service Modelling Execution Environment (WSMX) dar, welche sich hauptsächlich in drei Software-Komponenten unterteilt.

- **WSMX Integration API** stellt eine Sammlung von Bibliotheken zur Verfügung und wird für die Integration von lose gekoppelten Komponenten benötigt. Komponenten müssen die zur Verfügung stehenden Interfaces implementieren, um integriert werden zu können.
- **WSMX Core** ist eine Beispielimplementierung von WSMO und verwendet exemplarisch einige der „WSMX Integration API“-Schnittstellen.
- **WSMX Components** sind Funktionalitätem, die in die WSMX Core integriert werden können. Ein Beispiel einer Komponente ist die „RadexChorographie“, welche ein Teil der Funktionalität abbildet, die zur Komposition von Web-Diensten notwendig ist.

Für den praktischen Einsatz von WSMX ist ein Distributionspaket, in der aktuellen Version 0.4, verfügbar. Im Paket sind alle aufgeführten Komponenten enthalten und es wird dadurch die Möglichkeit geboten, die gesamten bereits implementierten Funktionen von WSMX nutzen zu können. Derzeit stehen drei Adapter zur Verfügung, um die Funktionen von WSMX zu nutzen. Durch den Start der Engine werden die Adapter initialisiert und es wird für jeden der Adapter ein Port geöffnet. Nach der Initialisierung können die Funktionen via SSH, HTTP oder per AXIS Web Service aufgerufen werden.
Die Abbildung 3.5 zeigt einen Ausschnitt des Erscheinungsbildes von WSMX bei der Verwendung des HTTP Adapters. Mit der aktuellen Implementierung ist nur eine Auführung der einzelnen Funktionen möglich, aber es wird nicht ermöglicht einen Prozess abarbeiten zu lassen, der z.B. eine komplette Anfrage darstellt.

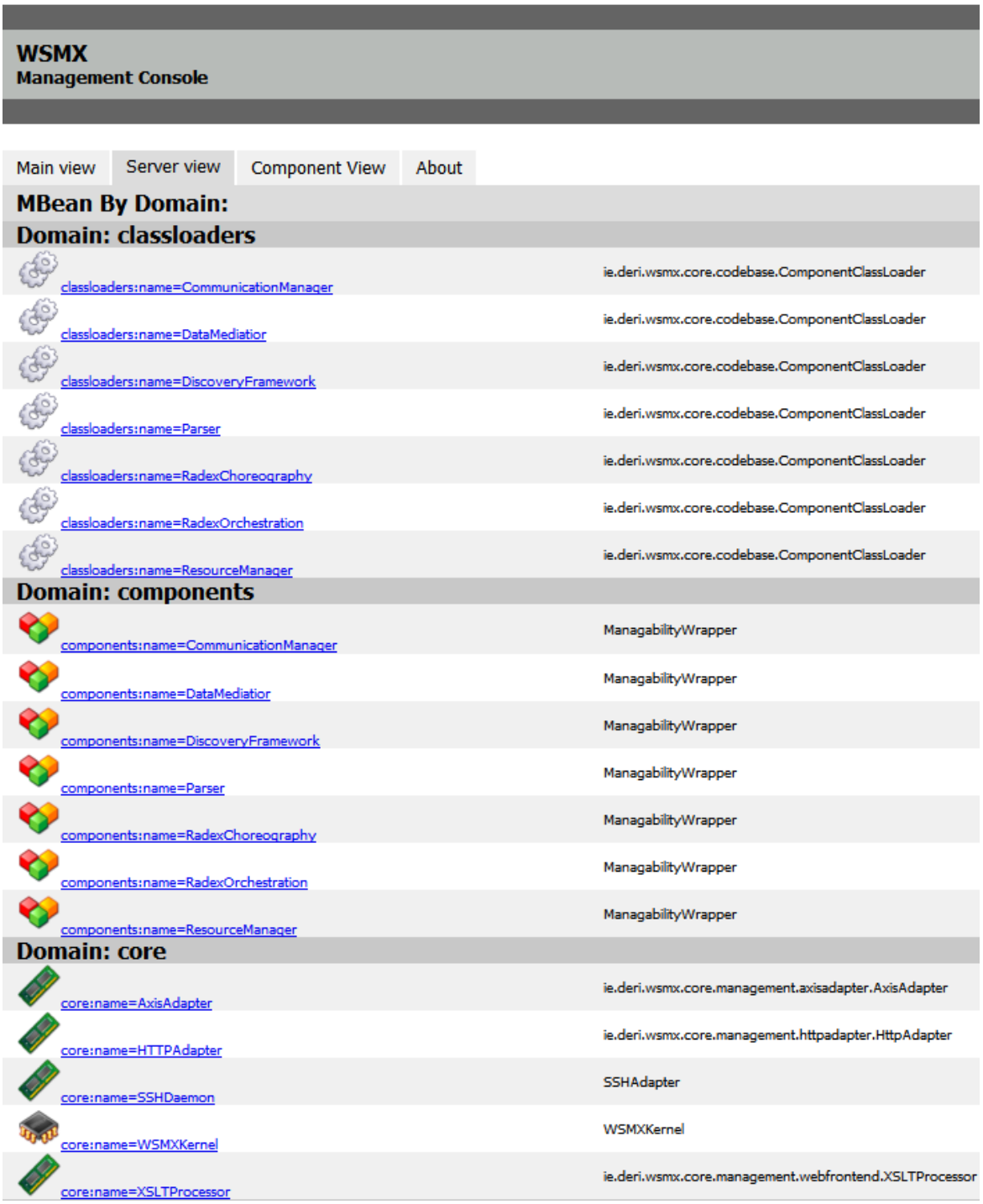

Abbildung 3.5: WSMX HTTP Interface

3.3 Vergleich von WSMX mit OWL-S

Eine Entwicklung, die vergleichbar mit WSMO ist, wird mit Web Ontology Language for Web Services (OWL-S) bezeichnet.
Der Aufbau von OWL-S stellt sich wie folgt dar.

- **Service Profile** – Das Service Profil enthält die Informationen über das Leistungsvermögen eines Dienstes, wie z.B. Vorbedingungen.
- **Service Model** – Das Service Model stellt ein Prozessmodell des Dienstes dar und beschreibt die funktionalen Eigenschaften des Web Service. Dieser Teil von OWL-S ist mit der WSMO Choreographie vergleichbar.
- **Service Grounding** – Die Informationen für eine Ausführung eines Dienstes werden durch das Service Grounding angegeben. Das Service Grounding bildet das abstrakte Prozessmodell als eine konkrete Service-Spezifikation ab. Es enthält z.B. Informationen über das Nachrichtenformat und das Netzwerkprotokoll.

Ein Hauptunterschied zwischen WSMO und OWL-S liegt darin, dass OWL-S sein Modell mit Hilfe der Web Ontologie Language (OWL) spezifiziert und WSMO dieses auf der Basis des Meta Object Facility (MOF) Modells durchführt. Das Meta Object Facility Modell ist ein aus vier Schichten bestehendes Metamodell (Fensel u. a., 2007).

- **Information** – Der unterste Informations-Layer beinhaltet die Daten, welche beschrieben werden sollen. Diese Ebene enthält die konkrete Angabe von Daten.
- **Model** – Der Model-Layer beschreibt die Meta-Daten der Elemente, die im Information-Layer enthalten sind. Zum Beispiel werden die Konzepte der Daten aus dem Informations-Layer angegeben.
- **Meta-Model** – Die Beschreibung der Struktur und Semantik der Meta-Daten des Model-Layers wird in dieser Ebene des Modells angegeben. Die Eigenschaften von WSMO werden in dieser Ebene beschrieben.
- **Meta-Meta-Model** – Die eigentlichen Sprachdefinitionen von WSMO sind im obersten Meta-Meta-Model Layer des Modells angelegt.

Im Gegensatz zu WSMO verwendet OWL-S zur Beschreibung seines Meta-Modells die gleiche Sprache wie zur Beschreibung der konkreten Daten. Es gibt keine klare Trennung der Schichten.
Einige weitere Unterschiede von WSMO und OWL-S sind im Folgenden aufgeführt. Eine ausführliche Auflistung ist in der Ausarbeitung (Lara u. a., 2005) zu finden.

- **Unterscheidung der Sicht eines Anbieters und Anfragenden** – OWL-S verwendet ein einziges Element zur Beschreibung eines Anbieters und Anfragenden. Im Gegensatz dazu wird bei WSMO zwischen Anfragendem (Goal) und Anbieter (Web Service) unterschieden.
- **Mediation** – WSMO ermöglicht eine lose Kopplung durch die Verwendung von Vermittlern. Es ist möglich, die Hauptelemente von WSMO miteinander zu verknüpfen und somit mögliche Probleme eines heterogenen Umfelds aufzulösen. Im Vergleich dazu besitzt OWL-S diese Eigenschaft nicht, da ein Vermittlungselement im Konzept von OWL-S nicht vorgesehen ist.
- **Nicht-funktionale Eigenschaften** – Für jedes Element von WSMO ist es möglich, nicht-funktionale Eigenschaften anzugeben. Im Gegensatz dazu wird bei OWL-S diese Möglichkeit nur bei der Definition von Profilen gegeben.

3.4 Bewertung

In der Einleitung dieses Kapitels wurde bereits kurz auf den Vorteil von WSMO, bezüglich der starken Fokussierung von anderen Projekten auf diese Technologie, eingegangen. Dieser Grund ist hauptsächlich ausschlaggebend für die Verwendung von WSMO in dieser Arbeit. Das IP SUPER Projekt (Kapitel 4) baut zum Beispiel auf WSMO auf und verwendet dieses zur Realisierung eines semantischen Geschäftsprozesses. Im Moment existiert keine vergleichbare andere Entwicklung in diesem Bereich.
Ein weiterer großer Vorteil von WSMO ist das ausgereifte Konzept, welches aber im Moment noch nicht vollständig realisiert ist. Im Vergleich zu WSMO ist das Konzept von OWL-S vollständig realisiert, was auf eine längere Entwicklungszeit zurückzuführen ist, aber es wird dennoch in aktuellen weiterführenden Projekten auf die Verwendung von OWL-S verzichtet.
Die Bewertung der Qualität des WSMO-Konzepts begründet sich weitestgehend auf den Vergleich mit OWL-S aus dem Abschnitt 3.3.

Trotz der Vorteile von WSMO ergeben sich einige Einschränkungen oder mögliche Probleme, die teilweise allgemein auf das semantische Umfeld, aber auch speziell auf WSMO zurückzuführen sind.

Die Basis für eine Modellierung von semantischen Prozessen ist das Bilden einer oder mehrerer Ontologien, welche die semantischen Relationen des Anwendungsumfelds strukturieren. Im Abschnitt 2.1 wurde bereits erwähnt, dass die Idee existiert, eine Ontologie von den jeweiligen Spezialisten des Bereichs erstellen zu lassen, um möglichst ein vollständiges und korrektes Resultat zu erzielen. In der Praxis sieht man sich dennoch mit dem Problem konfrontiert, dass eine Ontologie, selbst wenn diese durch einen Spezialisten erstellt wurde,

nicht als vollständig definiert anzusehen ist. Eine Begründung dafür ist, dass Wissen eine Perspektive, einen gewissen Grad an Subjektivität besitzt und nicht so einfach abgegrenzt werden kann.

> Everything is deeply intertwingled (Nelson, 1974).

Selbst in einem vollständig formalisierten Anwendungsgebiet spielt die Betrachtungsweise eine große Rolle und hat einen großen Einfluss auf das Resultat. Für den Bereich der semantischen Geschäftsprozesse lässt sich daraus ableiten, dass eine Fehlinterpretation möglich ist und nicht das gewünschte Ergebnis erzielt wird. Jede Änderung der Wissensbasis kann Einfluss auf das Ergebnis des Prozesses haben.
Ein weiterer relevanter Punkt bei der Verarbeitung von Prozessen ist die Laufzeit. Die Laufzeit eines technisch modellierten Prozesses kann ohne Weiteres bestimmt werden, wenn die Dienstgüte der einzelnen Komponenten vorliegt. Der Grund dafür ist, dass die verwendeten Komponenten bekannt sind und man den längsten Pfad des Prozesses bestimmen kann. Im Gegensatz dazu ist bei einem semantischen Prozess die Laufzeit nicht zu bestimmen, da die verwendeten Komponenten nicht bekannt sind und sich diese, je nachdem was für Eigenschaften eine Anfrage besitzt, ändern. Die Komponenten werden erst zur Laufzeit bestimmt (late binding) und es könnte eine Situation eintreten, dass zwei identische Anfragen unterschiedliche Komponenten verwenden. Die Situation, dass man nicht exakt festlegen kann, wie eine Abarbeitung durchgeführt werden soll, sorgt im ersten Moment vielleicht für Unbehagen, aber dadurch entstehen auch viele Vorteile, die genutzt werden können. Es wäre z.B. möglich, während des Prozessablaufs aus einer Sammlung von Diensten die Komponente zu bestimmen, die zum aktuellen Zeitpunkt die beste Dienstgüte besitzt. Das führt zu einer Verkürzung der Laufzeit und Verbesserung der Qualität des gesamten Prozesses.

Einige Einschränkungen, welche direkt auf WSMO bezogen werden können, sind im praktischen Umgang zu erkennen. Es handelt sich hierbei zwar um sehr spezielle Einschränkung oder nicht vorhandene Funktionalität, aber dennoch tauchen diese bei der Realisierung häufig auf.
Derzeit ist es z.B. nicht möglich, die Werte von zwei Eigenschaften eines Konzeptes, welche den Typ „String“ besitzen, zu konkatinieren. Ein Anwendungbeispiel für diese Funktionalität zeigt der Codeabschnitt 3.11.

Listing 3.11: Person-Konzept

```
concept Person
    displayName impliesType _string
    lastname impliesType _string
    firstname impliesType _string
```

```
instance pInstance memberOf Person
    lastname hasValue "Bernhardt"
    firstname hasValue "Robert"
```

Die Funktionalität, eine Konkatination durchzuführen, würde im Beispiel die Möglichkeit geben, die Eigenschaft „displayName“ aus den beiden Eigenschaften „lastname“ und „firstname“ zusammenzusetzen. Dieses könnte z.B. durch ein Axiom realisiert werden. In Java würde das gleiche Resultat durch den Code-Abschnitt „displayName = lastname + ’ ’ + firstname;“ ermöglicht werden.
Als Alternative zur Konkatination gibt es die Möglichkeit, einer Eigenschaft eine Menge zuzuweisen, was im Beispiel bedeuten würde, dass durch ein Axiom eine Menge aus den Werten von „lastname“ und „firstname“ an die Eigenschaft „displayName“ gebunden werden könnte. Trotz der Alternative handelt es sich bei der Zeichenketten-Verknüpfung um eine wünschenswerte Funktionalität.

Ein weiterer Punkt ist die Aufteilung von Diensten und ihrer Funktionen. Eine Eigenschaft von WSMO ist die Möglichkeit, in einer Definition eines Web Service mehrere Methoden semantisch zu beschreiben und auch die Informationen zur Ausführung zu hinterlegen. In der Praxis würde dieses bedeuten, dass ein Dienst mit allen Funktionen in einer Web Service-Beschreibung semantisch beschrieben wird. Das würde dazu führen, dass die Eigenschaften der Vor- und Nachbedingungen und die Komposition alle Informationen für alle Methoden des Dienstes beschreiben.
Im Konzept von WSMO ist vorgesehen, dass ein Dienst über eine Zieldefinition gefunden wird, was bei einer semantischen Web Service-Beschreibung mit mehreren Methoden dazu führen würde, dass ein Service gefunden wird, der nicht nur die geforderten, sondern auch weitere Funktionalitäten abdeckt. Für das Auffinden eines Dienstes stellt diese Eigenschaft kein Problem dar, aber bei der Verwendung einer Zieldefinition in einem Geschäftsprozess würde es bedeuten, dass eine Anfrage einen Einfluss auf die gewählte Funktionalität besitzt und somit ein definierter Prozess möglicherweise anders als vorgesehen ausgeführt wird.
Dieser Fall wird im Abschnitt 6.2 klar gezeigt, da dort der Einfluss einer Anfrage auf die konkrete Realisierung der gewünschten Funktionalität demonstriert wird.

4 Semantische Geschäftsprozesse

Wir haben in den vorangegangenen Abschnitten gesehen, dass ein Hinzufügen von semantischen Informationen den Vorteil besitzt, dass Maschinen diese Informationen verarbeiten können. Dieser Vorteil kann nun genutzt werden, um eine fachliche Komposition der Services in einem Geschäftsprozess, ohne eine technische Modellierung, vornehmen zu können.
Derzeit beschäftigt sich das Integrated Project SUPER Forschungsprojekt[1] mit der möglichen Realisierung dieser Funktionalität. Genau wie die WSMO Arbeitsgruppe, gehört das IP SUPER Projekt zu der European Semantic Systems Initiative (ESSI[2]).
Das IP SUPER Projekt stellt ein Framework dar, welches durch die Verwendung von verschiedenen Entwicklungen eine Ausführung eines semantischen Geschäftsprozesses ermöglicht. Die semantische Beschreibung der Komponenten eines Geschäftsprozesses erfolgt durch WSMO. Durch den konzeptionellen Ansatz eine Funktionalität, durch Definition eines Zieles, ausfinding zu machen, ist WSMO unter anderem besonders für die Verwendung innerhalb eines Geschäftsprozesses geeignet. Das begründet sich darauf, dass ein Element eines Prozesses mit einer Zieldefinition gleichgesetzt werden kann. In der Praxis wird jedes Element, welches eine Aufgabe darstellt, mit einem in WSMO modellierten Ziel verknüpft. Bei der Ausführung des Prozesses wird für jede Zieldefinition der entsprechende Dienst gesucht, welcher das gewünschte Leistungsvermögen besitzt.
Der fehlende Teil für die Ausführung eines semantischen Prozesses ist die Verknüpfung von einzelnen Aufgaben. Zu diesem Zweck wurde die Business Process Management Ontology (BPMO) entworfen. Dabei handelt es sich um eine Ontologie, welche die Eigenschaften eines Geschäftsprozesses enthält und miteinander in Relation setzt.
Durch Verwendung von WSMO und BPMO kann ein semantischer Geschäftsprozess mit Hilfe des IP SUPER Frameworks ausgeführt werden, was das Ziel der Arbeit darstellt. Der Ablauf einer Ausführung unterteilt sich in sechs Schritte und wird in der Abbildung 4.1 gezeigt.

[1] http://www.ip-super.org
[2] http://www.essi-cluster.org

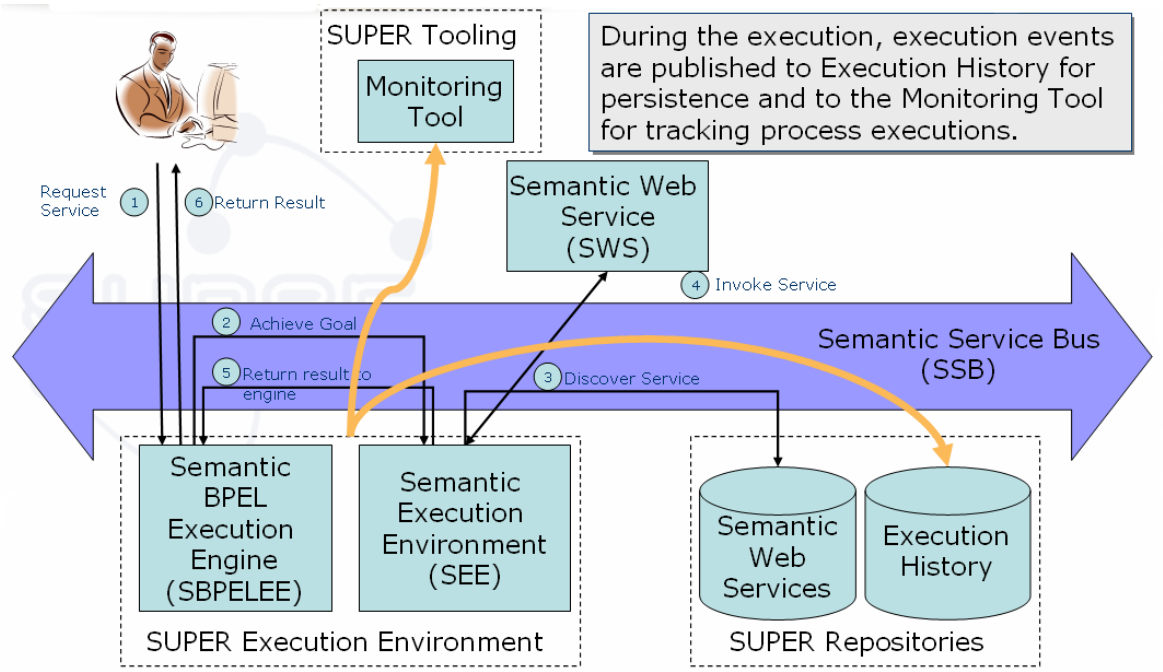

Abbildung 4.1: Ausführungs-Szenario (SUPER, 2007)

1. Request Service – Der Prozess wird mit einer Anfrage eines Users an die Semantic BPEL Execution Engine (SBPELEE) gestartet.
2. Achieve Goal – Die benötigte Funktion wird mit Hilfe der Übermittlung eines Goals an die Semantic Execution Environment (SEE) angefordert.
3. Discover Service – Ein Web Service, der die Funktionen zur Realisierung des Ziel erfüllt, wird in einem Service Repository gesucht.
4. Invoke Service – Die Ausführung des gefundenen Web Service führt die eigentliche Funktion aus und liefert ein Ergebnis.
5. Return Result to Engine – Das erhaltene Resultat der Ausführung des Goals wird an die Engine zurückgeliefert.
6. Return Result – Das letztendliche Ergebnis der Ausführung wird an den Benutzer übermittelt.

Zum aktuellen Zeitpunkt wird an einem Prototyp entwickelt, mit dessen Hilfe die aufgeführten Funktionen realisiert werden sollen. Das derzeitige Entwicklungsstadium ist nicht festzustellen, da keine Informationen darüber bekannt sind.
Der theoretische Ablauf einer Anfrage wird am Beispiel im Abschnitt 6.3 näher erläutert.

4.1 BPMN

Business Process Modeling Notation (BPMN) ist eine Spezifikation für eine grafische Sprache, die zur Abbildung von Geschäftsprozessen dient. BPMN enthält die Elemente, welche von BPMO verwendet werden und in der Ontologie abgebildet werden.
Bei der Spezifikation handelt es sich um ein Object Management Group[3] (OMG) Standard und sie bezieht sich auf die grafische Repräsentation von Prozessen. Die Elemente der Notation sind in vier Gruppen aufgeteilt.

- **Flow Objects** sind Elemente, die Knotenpunkte innerhalb eines Geschäftsprozesses definieren. Zu den möglichen Knotenpunkten zählen Activities, Gateways und Events.
- **Connecting Objects** verbinden Elemente eines Prozesses miteinander und es wird zwischen zwei Typen unterschieden. Sequence Flows definieren die Reihenfolge der Abarbeitung und Message Flow implizieren einen Nachrichtenaustausch zwischen den Elementen.
- **Swimlanes** – Eine Repräsentation der Teilnehmer an einem Prozess wird durch Pools ermöglicht, die sich durch Lane Elemente weiter unterteilen lassen.
- **Artifacts** ist eine Gruppe mit drei Elementen. Eine Text-Annotation ist mit einem Kommentar gleichzusetzen, ein Data Object ist ein Objekt, welches z.B. durch den Prozess bearbeitet wird, und eine Group ist ein Hilfsmittel, um Teile des Prozesses gruppieren zu können.

In der Abbildung 4.2 [4] ist ein Beispiel eines mit BPMN-Elementen modellierten Prozesses zu sehen.

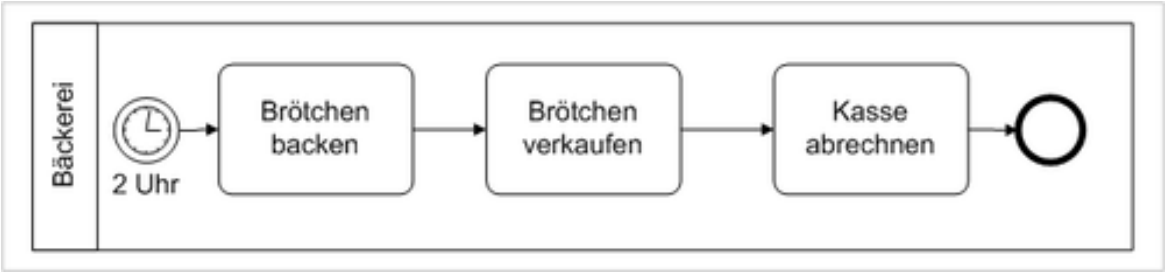

Abbildung 4.2: BPMN-Beispiel

[3] http://www.omg.org
[4] Quelle: http://de.wikipedia.org/wiki/BPMN

4.2 BPMO

Business Process Management Ontology (BPMO) ist eine Ontologie der BPMN-Elemente aus dem Abschnitt 4.1. Durch Bildung einer Prozess-Ontologie auf Basis der BPMO ist es möglich einen Geschäftsprozess anzugeben, dessen Ablauf und Funktionen durch die BPMO definiert werden.
Bei dem dargestellten Prozess in Abbildung 4.3 handelt es sich um die Visualisierung einer Ontologie.

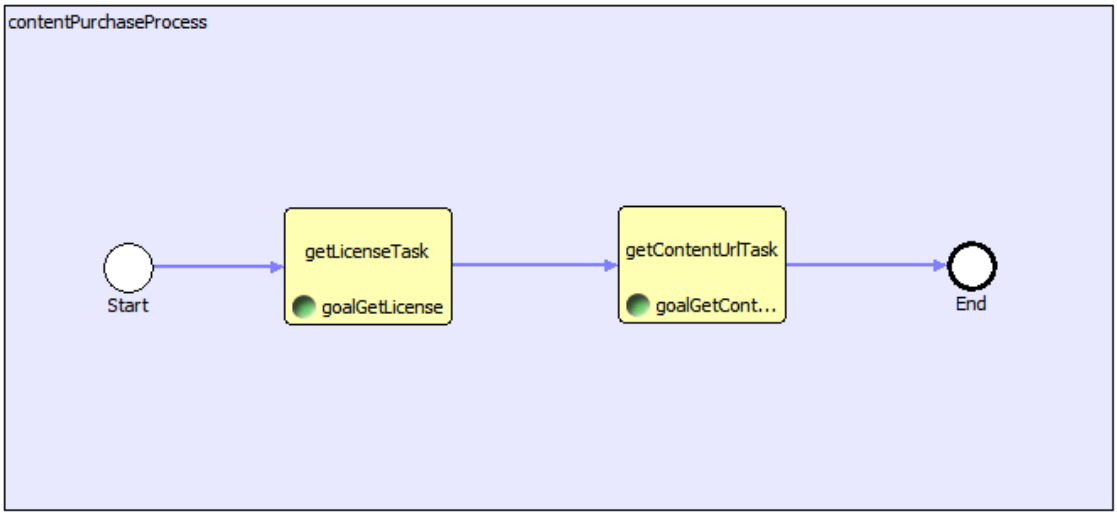

Abbildung 4.3: BPMO-Beispiel Prozess

5 Anwendungsbeispiel

Im Bereich des Kundenbeziehungsmanagements ist es wichtig Informationen von Kunden zu sammeln und diese miteinander in Verbindung zu setzen. Weiterhin spielt die Qualität der gesammelten Daten eine große Rolle, um ein aussagekräftiges Resultat zu erhalten. Im Folgenden werden Komponenten ausgewählt und entwickelt, die für mögliche Prozesse in diesem Bereich eine Rolle spielen.
Im Bereich des Kundenbeziehungsmanagements können semantische Geschäftsprozesse eine große Bedeutung bekommen, da eine Umsetzung mit einer technischen Modellierung eines Prozesses eine sehr hohe Komplexität besitzen kann. Bei einer Überprüfung von z.B. internationalen Adressen gibt es mehrere Wege, einen Dienst zur Verfügung zu stellen, welcher nicht die Beschränkung besitzt, nur Adressen aus einem bestimmten Land überprüfen zu können. Unter anderem wäre es möglich, einen Service bereitzustellen, der eine Choreografie durchführt und dabei externe Funktionen verwendet. Die Entscheidung über einen geeigneten Dienst würde durch eine bedingte Auswahl durchgeführt werden. Ein offensichtlicher Indikator für die Herkunft der Adresse stellt die Information des Landes dar und je nachdem welches Land angegeben wird, würde eine Auswahl der entsprechenden Dienste durchgeführt werden.

In der Abbildung 5.1 ist annähernd zu erkennen, welchen Umfang eine Entwicklung eines solchen Prozesses oder Web Service bedeuten würde. Bei einer Implementierung würde sicher auf eine Konfigurationsmöglichkeit geachtet werden, aber es müsste bei einer Änderung eines Dienstes oder bei einem Hinzufügen einer neuen Funktionalität eine Anpassung der Entwicklung stattfinden. Weiterhin können auf Grund der großen Anzahl und verteilten externen Komponenten Probleme auftreten. Zur Sicherstellung einer besseren Verfügbarkeit und Funktion müssen alternative Dienste angegeben werden, um eine Ausfallsicherheit und eine gute Service-Qualität zu gewährleisten.
Die technische Abbildung aller aufgezeigten Funktionalitäten würde zu einem sehr komplexen System führen und es wäre außerdem ein großer Aufwand, dieses System skalierbar zu gestalten.
Als gute Alternative zu einer herkömmlichen technischen Modellierung bietet sich ein semantischer Prozess an. Die Eigenschaften der Prozessteilnehmer werden semantisch angegeben und es würde, je nachdem was für eine Anfrage gestellt werden würde, ein geeigneter Dienst gesucht werden. In den folgenden Abschnitten wird diese Möglichkeit genauer erläutert.

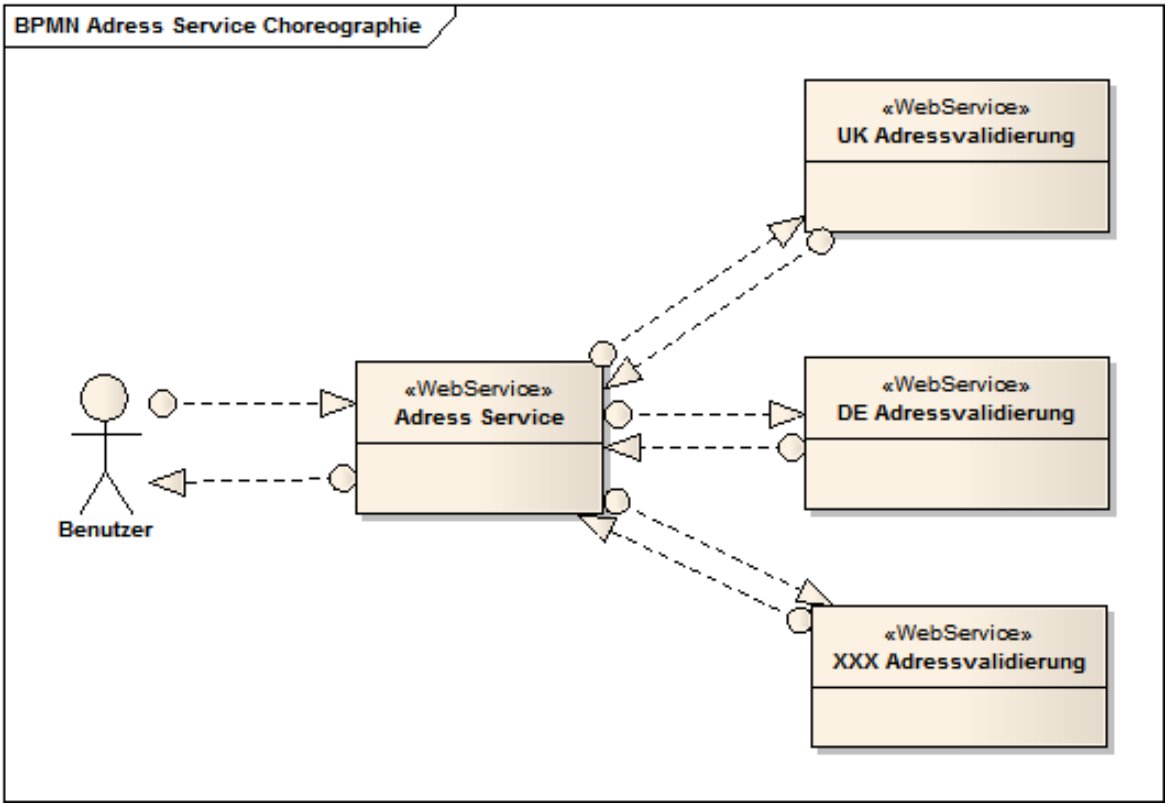

Abbildung 5.1: Choreografie zur Validierung von Adressdaten

5.1 Komponenten

In den folgenden drei Teilabschnitten werden Beispiele von Prozessteilnehmern aufgeführt und deren Eigenschaften beschrieben. Aus den aufgeführten Diensten wird im Anschluss ein Geschäftsprozess modelliert und deren Funktionsweise erläutert.

5.1.1 CAPTCHA Service

Bei einer CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) Validierung handelt es sich um eine Unterscheidung zwischen Mensch und Maschine. Diese Art der Validierung ist wichtig, damit keine Daten von anderen Software-Komponenten an das System gesendet werden können und somit Missbrauch vermieden wird.
Dem Benutzer wird ein Bild angezeigt, dessen genauen Inhalt er in ein Textfeld eingeben muss. Durch eine Übereinstimmungsprüfung des Bildinhalts und der Texteingabe wird eine

Validierung durchgeführt. Für eine Maschine wäre das Interpretieren des Bildinhalts schwierig, obwohl zum Beispiel die Optical Character Recognition (OCR) Technologie die Möglichkeit bietet, von Bildern Textinhalte zu extrahieren. Bei CAPTCHA-Bildern wird versucht, die Verwendung dieser Technologien zu verhindern, durch das gezielte Einsetzen von Bildstörungen.

Abbildung 5.2: Beispiel CAPTCHA-Bild

Bei dem Design einer Komponente zur Bereitstellung der CAPTCHA-Funktionalität ist auf den Ablauf aus Sicht des Benutzers zu achten. Daraus ergeben sich die benötigten Methoden.

1. Anzeige eines CAPTCHA-Bildes
2. Eingabe des CAPTCHA-Bildinhalts
3. Anzeige des Überprüfungsergebnisses

Aus dem beschriebenen Ablauf lässt sich ableiten, dass für die Bereitstellung drei Methoden benötigt werden. Während der Transaktion muss dem Service der Inhalt des Bildes bekannt sein, was durch einen eindeutigen Identifizierungschlüssel ermöglicht werden kann. Im Folgenden wird dafür die Bezeichnung CaptchaId verwendet, welche ein CAPTCHA-Bild genau identifiziert.

- GetRandom(int length): GUID

Parametername	**Pflichtangabe**	**Wertebereich**	**Beschreibung**
length	Ja	4 - 20	Anzahl der zufälligen Zeichen
Rückgabewert	—	GUID	Identifikationsschlüssel

Tabelle 5.1: CaptchaService > GetRandom > Parameter

- GetImage(GUID captchaId, ImageFormat: imageFormat, int width, int height): Byte()

Parametername	Pflichtangabe	Wertebereich	Beschreibung
captchaId	Ja	GUID	Identifikationsschlüssel
imageFormat	Ja	ENum: JPEG | PNG	Bildformat
width	Ja	100-300	Breite in Pixeln
height	Ja	30-300	Höhe in Pixeln
Rückgabewert	—	Byte()	Bilddaten

Tabelle 5.2: CaptchaService > GetImage > Parameter

- IsValid(GUID captchaId, String captchaValue): Boolean

Parametername	Pflichtangabe	Wertebereich	Beschreibung
captchaId	Ja	GUID	Identifikationsschlüssel
captchaValue	Ja	4-20 Zeichen	Text des Bildes
Rückgabewert	—	True | False	Aussage darüber, ob der eingegebene Wert dem des Bildes entspricht

Tabelle 5.3: CaptchaService > IsValid > Parameter

Wegen der schwerpunktmäßigen Verwendung von CAPTCHA-Diensten im Internet wird ein Adapter benötigt. Der Adapter macht es möglich, direkt eine URL anzugeben, um als Ergebnis das CAPTCHA-Bild zu erhalten.

5.1.2 Validierung von Adressdaten

In vielen Fällen ist es sinnvoll, eine Prüfung von Adressdaten bereits vor der Speicherung dieser durchzuführen. Sobald es sich z.B. um eine Bestellung handelt, bei der die Bezahlung per Rechnung durchgeführt werden kann, ist diese Validierung sehr wichtig, um Falscheingaben zu unterbinden und später eine Rechnung an diese Adresse senden zu können. Bei einer nachträglichen Überprüfung der Eingaben kann zwar eine Stornierung durchgeführt werden, wenn es sich um eine nicht korrekte Adresse handelt und keine Rechnung gestellt werden könnte, aber dies ist natürlich nicht im Interesse des Kunden und Unternehmens.
Eine Adresse kann als Erstes auf Vollständigkeit und Format geprüft werden. Als zweiter Schritt kann eine Überprüfung des Inhalts erfolgen. Der erste Schritt ist auch bei internationalen Adressen keine große Hürde, da dafür nur wenige Informationen für die Umsetzung benötigt werden. Bei einer Überprüfung des Inhalts muss dagegen auf eine große Menge an Daten zugegriffen werden und diese müssen möglichst immer aktuell sein. Es müssen

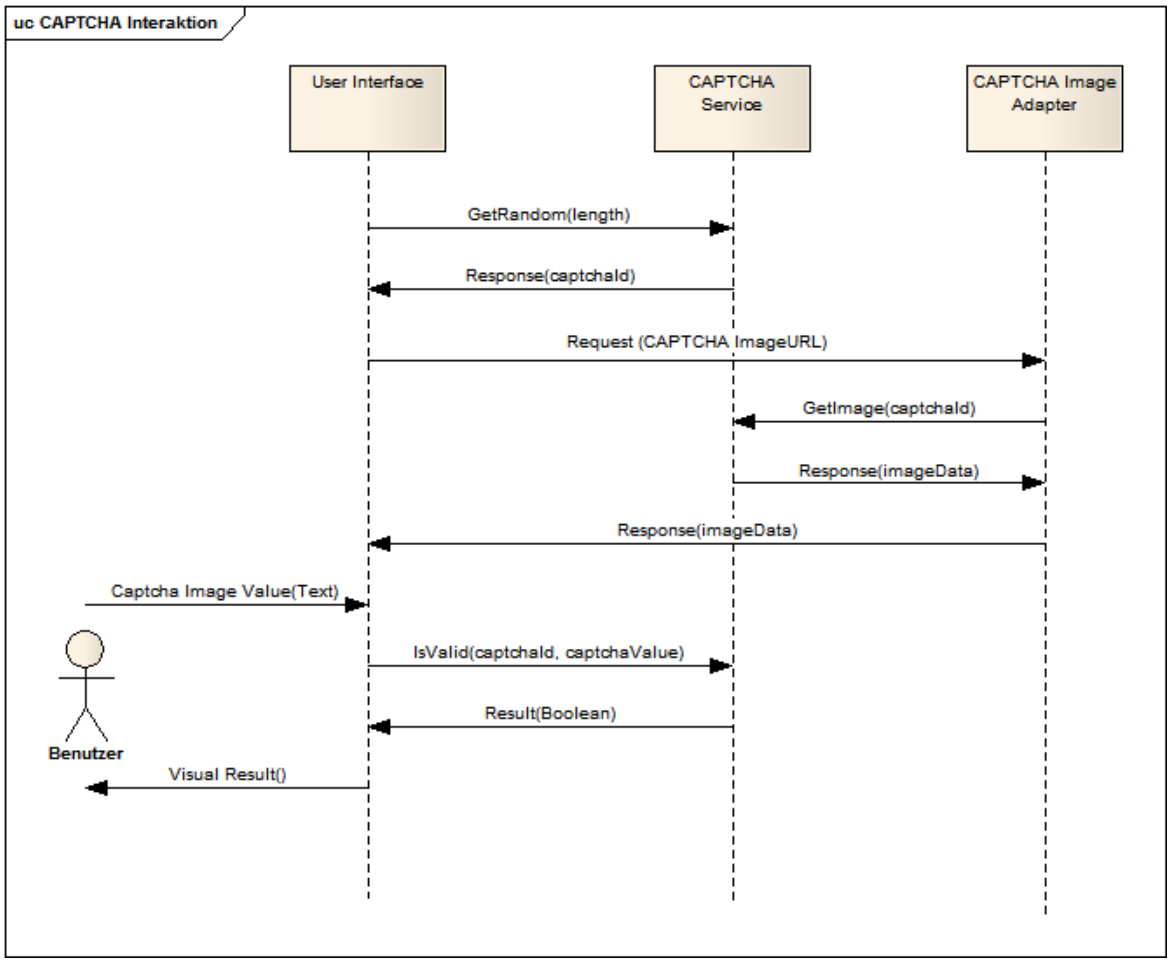

Abbildung 5.3: Interaktion – CAPTCHA Service

Informationen von Straßen, Postleitzahlen, Städten etc. vorhanden sein und ein Zusammenhang zwischen ihnen sollte gewährleistet sein, da ansonsten nur die einzelnen Elemente der Adresse geprüft werden können und das keine Aussage über die inhaltliche Korrektheit der Adresse zulässt.
Für den deutschen Raum bietet zum Beispiel die Deutsche Post einige Produkte, um diese Funktionalität abdecken zu können. In anderen Länder stehen ähnliche Produkte zur Verfügung und es werden unter anderem auch Real-Time-Anfragen ermöglicht.
Durch eine nicht vorhandene regionenübergreifende Standardisierung von Adressen werden je nachdem, auf welche Region sich die Adresse bezieht, zusätzliche Angaben und Formate verwendet. Dennoch finden sich bei Adressen aus unterschiedlichen Ländern Elemente, die eine gleiche oder ähnliche Bedeutung besitzen.

Die Tabellen 5.4 und 5.5 stellen nur die minimalen benötigten Angaben dar, damit eine Adres-

Adresselement	Pflichtangabe	Beschreibung
Mr A Smith	Wenn angebracht	Name
Acme Plc	Wenn angebracht	Unternehmen
Acme House	Ja (außer es besitzt eine Nummer)	Gebäude
3 High Street	Ja	Nummer und Straße des Gebäudes
Hedle End	Ja, wenn ein ähnlicher Straßenname in der Stadt existiert	Zusätzliche Informationen über die Lage
SOUTHAMPTON	Ja	Stadt/Ortschaft (Großbuchstaben)
SO31 4NG	Ja	Postleitzahl (Großbuchstaben)

Tabelle 5.4: Gültiges Adressformat (Großbritannien) – Quelle: Royal Mail

Adresselement	Pflichtangabe	Beschreibung
Herr Max Mustermann	Wenn angebracht	Name
MAMU GmbH	Wenn angebracht	Unternehmen/Organisation
Berliner Tor 5	Ja	Straße und Nummer des Gebäudes
20099 Hamburg	Ja	Postleitzahl und Stadt/Ort

Tabelle 5.5: Gültiges Adressformat (Deutschland)

se in dem jeweiligen Land als gültig angesehen wird. Bei einem Vergleich der Tabellen kann man einige Gemeinsamkeiten erkennen, aber dennoch liegen unterschiedliche Formatvorgaben vor. Zum Beispiel unterscheidet sich die Postleitzahl zwischen Deutschland und Großbritannien durch die Länge und die zulässigen Zeichen.
Für eine Demonstration der Möglichkeiten eines semantischen Prozesses werden im Folgenden zwei Dienste entwickelt, welche eine Überprüfung von Adressen aus Deutschland und Großbritannien durchführen können.

Großbritannien

Anhand der Tabelle 5.4 lassen sich die Parameter der benötigten Methode einfach identifizieren.

- IsValid(String name, String company, String building, String street, String streetNumber, String localityExtension, String city, String zipCode): Boolean

Parametername	Pflichtangabe	Wertebereich	Beschreibung
name	Nein, wenn „company“ angegeben wird	> 0 Zeichen	Name des Empfängers
company	Nein, wenn „name“ angegeben wird	> 0 Zeichen	Name des Unternehmens
building	Nein, wenn „streetNumber“ angegeben wird	> 0 Zeichen	Name des Gebäudes
streetNumber	Nein, wenn „building“ angegeben wird	> 0 Zeichen	Hausnummer
street	Ja	> 0 Zeichen	Name der Straße
localityExtension	Nein	> 0 Zeichen	Zusätzliche Informationen über die Lage
city	Ja	> 0 Zeichen	Name der Stadt/Ortschaft
zip	Ja	8 Zeichen	Postleitzahl
Rückgabewert	—	True | False	Aussage über das Ergebnis der Adressprüfung

Tabelle 5.6: AddressValidationService > Großbritannien > IsValid > Parameter

Bei einem Blick auf die Pflichtangeben ist leicht zu erkennen, dass keine offensichtliche Aussage über die Notwendigkeit der Angabe getätigt werden. Die Entscheidung, ob es sich um einen Pflichtparameter handelt, steht in einem Zusammenhang zu einem anderen Parameter. Ein Blick auf die technische Beschreibung des Service würde einem Anwender suggerieren, dass nur die Parameter „street“, „city“ und „zipCode“ benötigt werden. Die Ausführung einer Anfrage mit nur diesen drei Werten würde zu einem Fehlerfall führen.
Eine semantische Beschreibung des Web Service wird diesem Fehlerfall vorbeugen, da die Abhängigkeit zwischen den Parametern beschrieben werden kann.

Deutschland

Genau wie bei dem Web Service für die Überprüfung von Adressen aus Großbritannien, ermöglicht die Tabelle 5.5 eine leichte Identifizierung der benötigten Methoden-Parameter.

- IsValid(String name, String company, String street, String streetNumber, String zipCode, String city): Boolean

Parametername	Pflichtangabe	Wertebereich	Beschreibung
name	Nein, wenn „company“ angegeben wird	> 0 Zeichen	Name des Empfängers
company	Nein, wenn „name“ angegeben wird	> 0 Zeichen	Name des Unternehmens
streetNumber	Ja	> 0 Zeichen	Hausnummer
street	Ja	> 0 Zeichen	Name der Straße
zip	Ja	5 Zahlen	Postleitzahl
city	Ja	> 0 Zeichen	Name der Stadt/Ortschaft
Rückgabewert	—	True | False	Aussage über das Ergebnis der Adressprüfung

Tabelle 5.7: AddressValidationService > Deutschland > IsValid > Parameter

Auch bei diesem Web Service kann erst duch eine semantische Beschreibung sichergestellt werden, dass eine Anfrage alle Vorbedingungen erfüllt. Grund dafür ist der Zusammenhang zwischen den Parametern „name“ und „company“.

5.1.3 Speicherung von Kundendaten

Die Speicherung der Kundendaten sollte durch einen Dienst ermöglicht werden, der ein breites Spektrum an Daten aufnehmen kann und die Validierung der Daten gering priorisiert. Ein Grund dafür ist es zu ermöglichen auch Informationen zu speichern, die unter Umständen nicht vollständig oder korrekt sind. Diese Funktionalität wird bei Aktionen benötigt, bei denen der Anreiz für den Konsumenten nicht sehr groß ist, seine Daten anzugeben. Eine strikte Validierung der Daten würde eine mögliche Hemmschwelle bedeuten, die den Konsumenten dazu veranlassen könnte, seine Eingaben abzubrechen. Die Tatsache, dass durch eine nicht oder nur teilweise vorhandene Überprüfung der Eingaben fehlerhafte Daten gespeichert werden, kann akzeptiert werden, da es unter Umständen besser ist, nur teilweise korrekte Daten eines Konsumenten zu erhalten als überhaupt keine. Weiterhin kann durch eine nachträgliche Überprüfung und Anreicherung der Daten eine bessere Datenqualität gewährleistet werden.
Durch die Existenz eines Adressdaten-Validierungs-Service ist es für den Web Service zur Speicherung der Kundendaten nicht notwendig, eine strikte Validierung vorzunehmen. Sollte es notwendig sein, eine Validierung der Daten durchzuführen, würde eine Kombination der beiden Dienste diese Funktionalität abbilden.

Die Parameter des Web Service ergeben sich aus der Vereinigungsmenge der Adressdaten aus den Tabellen 5.4 und 5.5 und möglichen weiteren Kundeninformationen.

- AddAddress(String name, String company, String building, String street, String streetNumber, String zip, String city, String localityExtension, String country): GUID

Parametername	**Pflichtangabe**	**Wertebereich**	**Beschreibung**
name	Nein, wenn „company“ angegeben wird	> 0 Zeichen	Nachname
company	Nein, wenn „name“ angegeben wird	> 0 Zeichen	Unternehmen
building	Nein	> 0 Zeichen	Name des Gebäudes
street	Ja	> 0 Zeichen	Straße
streetNumber	Ja	> 0 Zeichen	Hausnummer
zip	Ja	> 0 Zeichen	Postleitzahl
city	Ja	> 0 Zeichen	Stadt
localityExtension	Nein	> 0 Zeichen	Zusätzliche Informationen über die Lage
country	Ja	> 0 Zeichen	Land
Rückgabewert	—	GUID	Identifikationsschlüssel

Tabelle 5.8: CustomerService > AddAddress > Parameter

5.2 Semantische Beschreibung der Komponenten

Der erste Schritt bei der semantischen Beschreibung von Komponenten ist die Identifizierung oder Erstellung von Ontologie-Konzepten. Dazu müssen die Bedeutungen von Eingabeparameter, Ausgabeparameter und der eigentlichen Funktionen bekannt sein. Diese Bedeutungen können durch eine Ontologie repräsentiert werden und später z.B. in Verbindung mit einem Web Service genutzt werden. Nach der Identifizierung der Elemente sollte eine grobe Einschätzung durchgeführt werden, ob die Möglichkeit besteht, dass die Bedeutung bereits in einer vorhandenen Ontologie erfasst wurde. Für Recherchezwecke bietet es sich an, eine Suchmaschine zu verwenden, bei der speziell nach Ontologie gesucht werden kann (z.B. `http://swoogle.umbc.edu`). Im Moment ist es noch schwer, geeignete Ontologie ausfindig zu machen, da viele bereitgestellte Ontologien nur zu Testzwecken erstellt wurden und unvollständig sind.
Der eigentliche Vorteil bei der Verwendung von externen Ontologien ist die Möglichkeit,

seine Wissensbasis erweitern zu können, ohne selbst aktiv Erweiterungen vorzunehmen. Dieser Vorteil kommt erst zum Tragen, wenn die verwendete Ontologie weit verbreitet ist und oft wiederverwendet wird. Diese Eigenschaft besitzen im Moment nur sehr wenige Ontologien, meistens im Umfeld der Entwicklung von semantischen Komponenten (z.B. http://www.wsmo.org/ontologies/dateTime/).
Für das aktuelle Anwendungsbeispiel wird auf eine Verwendung von externen Ontologien verzichtet. Ein nachträgliches Verwenden von externen Ontologien ist ohne Weiteres durch Mediatoren möglich.

5.2.1 Entwicklung der Konzepte

Eine praktikable Vorgehensweise für die Erstellung einer Ontologie ist das Identifizieren von benötigten Konzepten, deren Eigenschaften und den daraus resultierenden Instanzen. Bei der Erstellung von Konzepten sollte darauf geachtet werden, dass man Elemente identifiziert, die in einem anderen Kontext wiederverwendet werden können. Diese sinnvolle nicht funktionale Anforderung führt dazu, dass Konzepte vom Allgemeinen zum Speziellen modelliert werden sollten.
Für die Beschreibung der CAPTCHA-Funktionalität ergeben sich die Konzepte und Instanzen im Codeabschnitt 5.1. Das Captcha-Konzept konnte auf Grund seiner Eigenschaften von einem Image-Konzept abgeleitet werden, was zu einer möglichen Wiederverwendung des Image-Konzepts führt.

Listing 5.1: CAPTCHA-Konzepte

```
concept ImageFormat
    name impliesType _string

concept Image
    width impliesType _integer
    height impliesType _integer
    format impliesType ImageFormat

concept Captcha subConceptOf Image
    id impliesType _string
    value impliesType _string
    length impliesType _integer

concept CaptchaValidationResponse
    result impliesType _boolean

instance JPEG memberOf ImageFormat
```

```
18      name hasValue "JPEG"

20  instance PNG memberOf ImageFormat
        name hasValue "PNG"
22
    instance BMP memberOf ImageFormat
24      name hasValue "BMP"
```

Im Abschnitt 5.1.1 werden zwei Dienste beschrieben, die eine Adressvalidierung durchführen und bei denen es wichtig ist, aus welchem Land die zu validierende Adresse stammt. Aus diesem Grund wird die nachfolgende Standort-Ontologie gebildet.

Listing 5.2: Standort-Konzepte

```
    concept Country
2       name impliesType _string

4   instance Germany memberOf Country
        name hasValue "Germany"
6
    instance Greatbritain memberOf Country
8       name hasValue "Greatbritain"

10  concept City
        name impliesType _string
12      country impliesType Country
        zip impliesType _string
14
    instance Hamburg memberOf City
16      name hasValue "Hamburg"
        country hasValue Germany
18      zip hasValue {"22087", "20097"}

20  instance London memberOf City
        name hasValue "London"
22      country hasValue Greatbritain
        zip hasValue {"EC1N-8JH", "NW5-2TJ"}
```

Für die Abbildung der Funktionalität der beiden beschriebenen Web Services aus Abschnitt 5.1.1 und 5.1.2 werden die nachfolgenden Konzepte benötigt. Eine Besonderheit, im Vergleich zu den vorigen Ontologien, ist die Verwendung von zwei Axiomen (cityByZIPCode

und countryByCity). Durch diese beiden gebildeten Axiome kann eine Adressinstanz automatisch erweitert werden. Bei der Angabe einer Postleitzahl und dem Land wird automatisch die entsprechende Stadt erkannt, wenn dieses in der Standort-Ontologie 5.2 mit den benötigten Informationen angegeben wurde. Diese Funktion wird durch das Axiom „cityByZIPandCountry“ ermöglicht. Das weitere Axiom „countryByCity“ ermittelt das Land zu einer entsprechenden Stadt.

Listing 5.3: Adress-Konzepte

```
axiom cityByZIPandCountry
    definedBy
        ?address[
            zip hasValue ?zip,
            loc#country hasValue ?country
        ] memberOf Address
        and ?city[
            loc#zip hasValue ?zip,
            loc#country hasValue ?country
        ] memberOf loc#City
        implies ?address[
            city hasValue ?city
        ].

axiom countryByCity
    definedBy
        ?address[
            city hasValue ?city
        ] memberOf Address
        and ?city[loc#country hasValue ?country] memberOf loc#City
        implies ?address[
            country hasValue ?country
        ].

concept Address
    name impliesType _string
    company impliesType _string
    street impliesType _string
    streetNumber impliesType _string
    localityExtension impliesType _string
    building impliesType _string
    zip impliesType _string
    city impliesType loc#City
```

```
    country impliesType loc#Country
35
concept AddressValidationResponse
37    result impliesType _boolean
    message impliesType _string
39
concept SaveAddressResponse
41    result impliesType _boolean
```

5.2.2 Web Services

Durch die semantische Beschreibung eines Web Service ist es möglich, die Vorbedingungen bereits vor der Ausführung zu prüfen. Das gilt sowohl für die geforderte Formatierung als auch für die Bedeutung. Sollte die Anfrage nicht den Vorgaben des Service entsprechen, würde dieser nicht ausgewählt werden und eine fehlerhafte Ausführung würde vermieden werden. In diesem Zusammenhang wird als Vorbedingung nicht die Eigenschaft „precondition“ bezeichnet, sondern „transitionRules“. Die Eigenschaft „precondition“ dient ausschließlich zur Identifizierung durch eine Zieldefinition, aber wird nicht im Zusammenhang mit einer Anfrage verwendet. Zur Definition der Vorbedingung von Anfrageparametern kann die Eigenschaft „transitionRules“ verwendet werden.

CAPTCHA Validation

Die semantische Beschreibung des CAPTCHA Dienstes wird im Listing 5.4 aufgeführt. Eine Besonderheit der Beschreibung ist die Begrenzung auf eine Methode (IsValid) des CAPTCHA Service, da durch deklaration von mehreren Methoden eines Dienstes, eine Anfrage Einfluss auf die Verwendung des Endpunktes hätte. Eine genaue Erläuterung wurde bereits im Abschnitt 3.4 gegeben.

Listing 5.4: Captcha Service Capability

```
1 postcondition
    definedBy
3       ?response[
            glb#result hasValue ?result
5       ] memberOf glb#CaptchaValidationResponse.

7 interface IsValidInterface
    choreography
```

```
        stateSignature
            importsOntology {
                _"http://www.newsarea.de/wsmo/ontologies/global/v1
                    .0/Global.wsml"
            }

            in
                concept glb#Captcha withGrounding _"http://services
                    .newsarea.de/captcha/v1.0/CaptchaService.asmx?
                    WSDL#wsdl.interfaceMessageReference(
                    CaptchaServiceSoap/IsValid/In)"
            out
                concept glb#CaptchaValidationResponse withGrounding
                    _"http://services.newsarea.de/captcha/v1.0/
                    CaptchaService.asmx?WSDL#wsdl.
                    interfaceMessageReference(CaptchaServiceSoap/
                    IsValid/Out)"

    transitionRules
        forall {?request} with
            (?request[
                glb#id hasValue ?id,
                glb#value hasValue ?value
            ] memberOf glb#Captcha)
        do
            add(_#1 memberOf glb#CaptchaValidationResponse)
        endForall
```

AdressValidation Web Service

Der Hauptunterschied beider nachfolgenden semantischen Dienstbeschreibungen ist der Wert der Eigenschaft „country“, innerhalb der „transitionRules“ Definition. In Abschnitt 6 wird gezeigt, welche Auswirkung diese Eigenschaft besitzt und inwieweit eine Anfrage einen Einfluss auf die Auswahl des Dienstes besitzt.

Listing 5.5: AddressValidation DE Service Capability

```
postcondition
    definedBy
        ?response[
```

```
            glb#result hasValue ?result,
            glb#message hasValue ?message
        ] memberOf glb#AddressValidationResponse.

interface IsValidInterface
    choreography
        stateSignature
            importsOntology {
                _"http://www.newsarea.de/wsmo/ontologies/global/v1
                    .0/Global.wsml"
            }

            in
                concept glb#Address withGrounding _"http://services
                    .newsarea.de/addressvalidation/de/v1.0/
                    AddressValidationService.asmx?WSDL#wsdl.
                    interfaceMessageReference(
                    AddressValidationServiceSoap/IsValid/In)"
            out
                concept glb#AddressValidationResponse withGrounding
                    _"http://services.newsarea.de/addressvalidation
                    /de/v1.0/AddressValidationService.asmx?WSDL#wsdl
                    .interfaceMessageReference(
                    AddressValidationServiceSoap/IsValid/Out)"

    transitionRules
        forall {?request} with
            (?request[
                glb#street hasValue ?street,
                glb#streetNumber hasValue ?streetNumber,
                glb#zip hasValue ?zip,
                glb#city hasValue ?city,
                glb#country hasValue loc#Germany
            ] memberOf glb#Address)
        do
            add(_#1 memberOf glb#AddressValidationResponse)
        endForall
```

Listing 5.6: AddressValidation UK Service Capability

```
capability AddressValidationServiceCapability
```

```
postcondition
    definedBy
        ?response[
            glb#result hasValue ?result,
            glb#message hasValue ?message
        ] memberOf glb#AddressValidationResponse.

interface IsValidInterface
    choreography
        stateSignature
            importsOntology {
                _"http://www.newsarea.de/wsmo/ontologies/global/v1
                    .0/Global.wsml"
            }

            in
                concept glb#Address withGrounding _"http://services
                    .newsarea.de/addressvalidation/uk/v1.0/
                    AddressValidationService.asmx?WSDL#wsdl.
                    interfaceMessageReference(
                    AddressValidationServiceSoap/IsValid/In)"
            out
                concept glb#AddressValidationResponse withGrounding
                    _"http://services.newsarea.de/addressvalidation
                    /uk/v1.0/AddressValidationService.asmx?WSDL#wsdl
                    .interfaceMessageReference(
                    AddressValidationServiceSoap/IsValid/Out)"

    transitionRules
        forall {?request} with
            (?request[
                glb#street hasValue ?street,
                glb#streetNumber hasValue ?streetNumber,
                glb#zip hasValue ?zip,
                glb#city hasValue ?city,
                glb#country hasValue loc#Greatbritain
            ] memberOf glb#Address)
        do
            add(_#1 memberOf glb#AddressValidationResponse)
```

```
33          endForall
```

SaveAddress Web Service

Das Listing 5.7 zeigt die Dienstbeschreibung eines Service zur Speicherung von Adressdaten.

Listing 5.7: SaveAddress Service Capability

```
1  postcondition
       definedBy
3          ?response memberOf glb#SaveAddressResponse.

5  interface SaveInterface
       choreography
7          stateSignature
               importsOntology {
9                  _"http://www.newsarea.de/wsmo/ontologies/global/v1
                       .0/Global.wsml"
               }
11
               in
13                 concept glb#Address withGrounding _"http://services
                       .newsarea.de/saveaddress/v1.0/SaveAddressService
                       .asmx?WSDL#wsdl.interfaceMessageReference(
                       SaveAddressServiceSoap/Save/In)"
               out
15                 concept glb#SaveAddressResponse withGrounding _"
                       http://services.newsarea.de/saveaddress/v1.0/
                       SaveAddressService.asmx?WSDL#wsdl.
                       interfaceMessageReference(SaveAddressServiceSoap
                       /Save/Out)"

17     transitionRules
           forall {?request} with
19             (?request[
                   glb#street hasValue ?street,
21                 glb#streetNumber hasValue ?streetNumber,
                   glb#zip hasValue ?zip,
23                 glb#city hasValue ?city,
```

```
                glb#country hasValue ?country
            ] memberOf glb#Address)
        do
            add(_#1 memberOf glb#SaveAddressResponse)
        endForall
```

5.2.3 Goals

Die Zieldefinition für die Auffindung der entsprechenden Dienste beschränkt sich in dem Beispiel auf die Nachbedingung. Die Dienste werden über die Eigenschaft der Rückgabe gefunden. Die Validierungsfunktion des CAPTCHA Service liefert z.B. das Resultat der Überprüfung „CaptchaValidationResponse“. Diese Information wurde in der Beschreibung der Web Services angegeben.
Die Verwendung von anderen Bedingungen ist in diesem Anwendungsbeispiel nicht notwendig, da eine eindeutige Auswahl des Dienstes nicht über eine Zieldefinition (Goal) durchgeführt wird, sondern über die Informationen der Benutzeranfragen.
Im Anwendungsbeispiel dient ein Goal nur zur groben Auswahl von Diensten, die die benötigte Funktionalität anbieten.

Listing 5.8: CAPTCHAValidation – Postcondition

```
postcondition
    definedBy
        ?response memberOf glb#CaptchaValidationResponse.
```

Listing 5.9: AddressValidation – Postcondition

```
postcondition
    definedBy
        ?response memberOf glb#AddressValidationResponse.
```

Listing 5.10: SaveAddress – Postcondition

```
postcondition
    definedBy
        ?response memberOf glb#SaveAddressResponse.
```

6 Demonstration

Die Demonstration gliedert sich in zwei Abschnitte. Im ersten Abschnitt wird das Auffinden eines einzelnen Web Service demonstriert und im zweiten Abschnitt wird ein Beispiel eines Geschäftsprozesses ausgeführt.

6.1 Software Komponenten

Die verwendeten Komponenten beruhen in beiden Abschnitten auf unterschiedlichen Entwicklungen. Der Grund dafür ist eine noch nicht vorhandene Implementierung des IP SUPER Konzepts.
Für das Auffinden eines Web Service mit Hilfe einer Zieldefinition (Goal) wird die WSMX Engine (Version 0.4) verwendet, welche bereits die dafür benötigten Funktionen besitzt. Nähere Informationen zur WSMX Engine wurden im Abschnitt 3.2.6 gegeben.
Die Ausführung eines Beispielprozesses wird im Abschnitt 6.3 mit Hilfe des Internet Reasoning Service (IRS-III) ermöglicht. Nähere Informationen zum IRS-III sind auf der Internet Seite zu finden (http://kmi.open.ac.uk/projects/irs/).

6.2 Web Service Auffindung

Für die Auffindung eines Web Service anhand eines Ziels und einer Anfrage können die Komponenten der WSMX verwendet werden. Der gesamte Prozess der Auffindung ist in zwei Teile aufgeteilt. Im ersten Schritt wird ein geeigneter Dienst ausfindig gemacht, welcher die vom Ziel angegebenen Funktionalitäten unterstützt. Der zweite Schritt überprüft die Möglichkeit, ob einer der gefundenen Dienste die Anfrage verarbeiten kann.
Für ein Beispiel wird die Suche nach einem AdressValidierung Service durchgeführt. Der zu suchende Dienst muss den Vorgaben des Ziels 5.9 entsprechen und die Benutzer-Anfrage verarbeiten können.
WSMO stellt für die Auffindung eines Dienstes eine „DiscoveryFramework“ Komponente zur Verfügung (siehe Abbildung 6.1). Auf alle diese WSMO-Komponenten kann auf unterschiedliche Weise zugegriffen werden. Es besteht zum Beispiel die Möglichkeit, die Funktionen der

Engine über ein HTTP Interface (siehe Abbildung 6.1), eine SSH Console oder eine Web Service Schnittstelle aufzurufen.

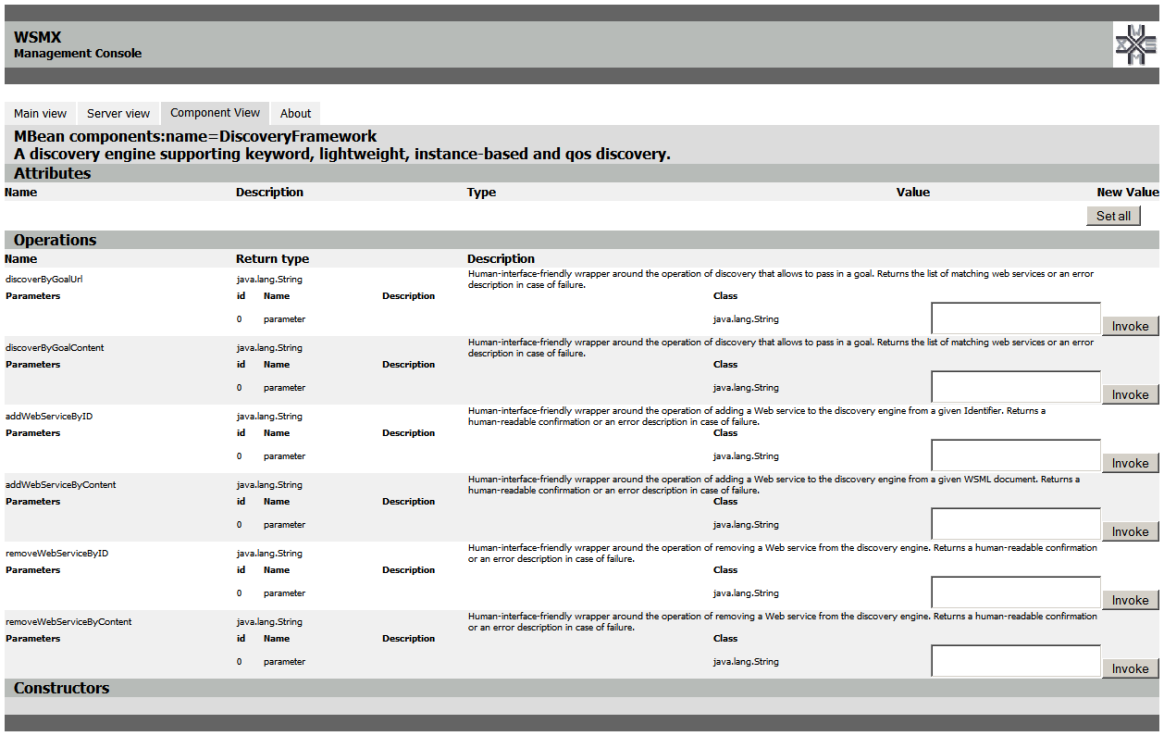

Abbildung 6.1: WSMX Discovery Framework

Für die Demonstration des Beispiels wurden die Quelldateien verwendet und die Funktionen direkt aufgerufen. Der Grund dafür sind Funktionsstörungen im Umgang mit den kompilierten Dateien. Bevor eine Ausführung wie gewünscht durchgeführt werden konnte, mussten einige Anpassungen im Quellcode durchgeführt werden. Ein Beispiel für eine Veränderung ist die Anpassung der Auffindungsarten. Standardmäßig wird bei WSMX eine Schlüsselwortsuche durchgeführt, bevor die eigentliche Auffindung über die Bedingungen erfolgt. Die Verwendung der Schlüsselwortsuche hat den Vorteil, dass die Gesamtdauer der Anfrage minimiert wird, da nicht alle Dienste über ihre Bedingungen identifiziert werden, sondern in erster Line durch eine Volltextsuche innerhalb dieser.
Im derzeitigen Stadium der Entwicklung ist die Schlüsselwortsuche noch nicht vollständig funktionsfähig und es kommt zu Problemem bei der Ausführung. Einer der Gründe dafür ist, dass ein Ziel nicht die gleichen Zeichenketten enthalten muss wie ein Web Service, obwohl der Service bei einer Identifizierung über die Bedingung die benötigten Funktionen erfüllen würde. Aus diesem Grund musste die „KEYWORD DISCOVERY“ Funktion ausgeschaltet werden.

Weiterhin ist es derzeit nicht möglich, über die Benutzerschnittstellen eine Anfrage an einen Service zu simulieren und damit festzustellen, ob die Anfrage von dem gewählten Service bearbeitet werden kann.

In den beiden folgenden Abschnitten werden, wie bereits erwähnt, zwei Anfragen mit unterschiedlichen Eigenschaften gestellt, wobei die Identifizierung über das Ziel gleich bleibt. Damit soll demonstriert werden, wie eine Anfrage Auswirkungen auf die Wahl des geeigneten Dienstes hat.

6.2.1 Anfrage mit einer Adresse aus Deutschland

Die Instanz 6.1 stellt eine Adresse aus Deutschland dar und wird in Verbindung mit der Zieldefinition 5.9 zur Auffindung eines Dienstes verwendet.

Listing 6.1: AddressValidation – Anfrage mit Adresse aus Deutschland

```
instance addr memberOf glb#Address
    glb#street hasValue "Lübecker Str."
    glb#streetNumber hasValue "25b"
    glb#zip hasValue "22087"
    glb#city hasValue loc#Hamburg
    glb#country hasValue loc#Germany
```

Im ersten Schritt werden die zur Verfügung stehenden Dienste in die Liste der zu durchsuchenden aufgenommen. Der zweite Schritt verwendet eine Instanz mit dem Parameter der Anfrage als Inhalt und das Ziel zur groben Auffindung eines geeigneten Dienstes.
Das Resultat der Suche liefert eine Auflistung der WSML-Dateien, welche die Beschreibung der Dienste enthalten, die die gewünschte Funktionalität der Zieldefinition (Goal) 5.9 abbilden. Im Beispiel werden Dienste gefunden, die eine Adressvalidierung durchführen können. Im letzten Schritt wird die Eignung der Services geprüft, die Anfrage verarbeiten zu können. Diese Überprüfung erfolgt auf der Basis der „transitionRules“ der Web Services 5.2.2.
Das Resultat zeigt, dass der erste Dienst die Anfrage verarbeiten kann. Durch Auslesen der Web Service Beschreibung ist es möglich, den Endpunkt zu identifizieren, der das Ausführen der Funktionalität ermöglicht. Im Listing 6.2 wird der entsprechende Endpunkt als „Grounding“ ausgewiesen.

Listing 6.2: AddressValidation – Ausgabe

```
Get Web Service by Goal
Append WS: AddressValidationService_DE.wsml
Successfully added Web service to LightweightDiscovery engine

```

```
Append WS: AddressValidationService_UK.wsml
Successfully added Web service to LightweightDiscovery engine
---
Append WS: CaptchaService_GetImage.wsml
Successfully added Web service to LightweightDiscovery engine
---
Append WS: CaptchaService_IsValid.wsml
Successfully added Web service to LightweightDiscovery engine
---
Append WS: SaveAddressService.wsml
Successfully added Web service to LightweightDiscovery engine
---
http://www.newsarea.de/wsmo/webservices/addressvalidation/
    AddressValidationService_DE.wsml
http://www.newsarea.de/wsmo/webservices/addressvalidation/
    AddressValidationService_UK.wsml

Selected Result:
http://www.newsarea.de/wsmo/webservices/addressvalidation/
    AddressValidationService_DE.wsml

Get Rounding:
Grounding: http://services.newsarea.de/addressvalidation/de/v1.0/
    AddressValidationService.asmx?WSDL#wsdl.
    interfaceMessageReference(AddressValidationServiceSoap/IsValid/
    In)
```

6.2.2 Anfrage mit einer Adresse aus Großbritannien

Die nachfolgende Instanz stellt eine Adresse aus Großbritannien dar. Der signifikante Unterschied zur Instanz aus Abschnitt 6.1 ist die Angabe eines anderen Landes („loc:Greatbritain“). Durch diese Änderung wird auch das Ergebnis verändert, da nun ein Dienst gewählt werden muss, der die angegebene Adresse verarbeiten kann.

Listing 6.3: AddressValidation – Anfrage mit Adresse aus Großbritannien

```
instance addr memberOf glb#Address
    glb#streetNumber hasValue "23"
    glb#street hasValue "Belgrave Square"
    glb#zip hasValue "SW1X 8PZ"
```

```
    glb#city hasValue loc#London
    glb#country hasValue loc#Greatbritain
```

Im Vergleich zu der Ausgabe im letzten Abschnitt ändert sich nur der letzte Teil. Die gelieferten Dienste wurden weiterhin über das gleiche Ziel gefunden, nur die Überprüfung der Möglichkeit, ob der Dienst die Anfrage bearbeiten kann, unterscheidet sich darin, dass erst beim zweiten Service die benötigte Funktionalität festgestellt werden konnte. Aus diesem Grund ist der Endpunkt ein anderer.

Listing 6.4: AddressValidation – Ausgabe

```
Get Web Service by Goal
Append WS: AddressValidationService_DE.wsml
Successfully added Web service to LightweightDiscovery engine
---
Append WS: AddressValidationService_UK.wsml
Successfully added Web service to LightweightDiscovery engine
---
Append WS: CaptchaService_GetImage.wsml
Successfully added Web service to LightweightDiscovery engine
---
Append WS: CaptchaService_IsValid.wsml
Successfully added Web service to LightweightDiscovery engine
---
Append WS: SaveAddressService.wsml
Successfully added Web service to LightweightDiscovery engine
---
http://www.newsarea.de/wsmo/webservices/addressvalidation/
    AddressValidationService_DE.wsml
http://www.newsarea.de/wsmo/webservices/addressvalidation/
    AddressValidationService_UK.wsml

Selected Result:
http://www.newsarea.de/wsmo/webservices/addressvalidation/
    AddressValidationService_DE.wsml

Get Rounding:
---


Selected Result:
http://www.newsarea.de/wsmo/webservices/addressvalidation/
    AddressValidationService_UK.wsml
```

```
28
   Get Rounding:
30 Grounding: http://services.newsarea.de/addressvalidation/uk/v1.0/
       AddressValidationService.asmx?WSDL#wsdl.
       interfaceMessageReference(AddressValidationServiceSoap/IsValid/
       In)
```

Die gezeigte Demonstration einer Suche nach einem Dienst kann als ein Element eines Geschäftsprozesses angesehen werden. In einem Geschäftsprozess werden für jedes Element die gleichen Prinzipien verwendet, um einen Endpunkt zu finden, an den die Daten der Anfrage gesendet werden können.

6.3 Ausführung eines Beispielprozesses

Durch die Kombination der Arbeitsschritte aus dem beschriebenen Anwendungsfall 5 kann ein Geschäftsprozess modelliert werden. Der Prozess in Abbildung 6.2 führt eine CAPTCHA und Adressdaten-Validierung aus und speichert danach die angegebene Adresse.

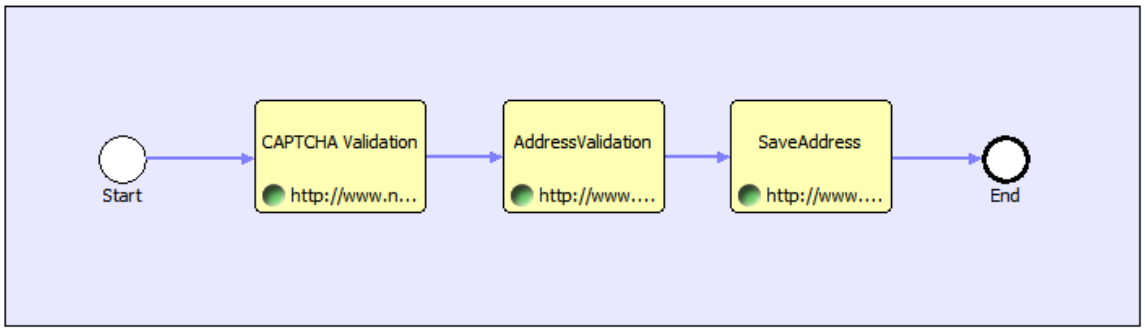

Abbildung 6.2: BPMO-Prozess des Anwendungsbeispiels

Derzeit ist es nicht möglich, die Ausführung eines eigenen Prozesses zu zeigen, da die dafür benötigten Komponenten noch nicht zur Verfügung stehen. Aus diesem Grund wird im Folgenden nur der theoretische Ablauf einer Anfrage beschrieben.
Im Kapitel 4 wurde der Ablauf einer Anfrage in Bezug auf die Konzepte des IP SUPER Projekts dargestellt. Auf der Basis dieses Ablaufs soll im Folgenden eine mögliche theoretische Ausführung des in der Abbildung 6.2 gezeigten Geschäftsprozesses demonstriert werden.

Listing 6.5: IP SUPER – Anfrage

```
instance captcha memberOf glb#Captcha
    glb#id hasValue "8f7df445-c634-4dd8-a62f-49736cd66050"
    glb#value hasValue "d2aSd2"

instance addr memberOf glb#Address
    glb#street hasValue "Lübecker Str."
    glb#streetNumber hasValue "25b"
    glb#zip hasValue "22087"
    glb#city hasValue loc#Hamburg
    glb#country hasValue loc#Germany
```

1. Request Service – Der Prozess wird mit der Anfrage (Listing 6.5) gestartet. In der Anfrage sind alle benötigten Informationen enthalten, um den kompletten Prozess abarbeiten zu können. Der Beispielprozess benötigt die Daten für die CAPTCHA Validierung und die zu validierende und speichernde Adresse.
2. Achieve Goal (CAPTCHA Validierung) – Der erste Prozessschritt stellt die CAPTCHA Validierung dar (Abbildung 6.2) und ist mit einer Zieldefinition (Listing 5.8) verknüpft. Zusammen mit der Anfrage wird die Zieldefinition (Goal) an die Semantic Execution Environment (SEE) gesendet.
3. Discover Service (CAPTCHA Validierung) – Die Semantic Execution Environment sucht, anhand der übermittelten Zieldefinition, (CAPTCHA Validation) einen Dienst im Repository, welcher die benötigten Fähigkeiten besitzt. Im Beispiel würde der Web Service aus Abschnitt 5.1.1 gefunden werden.
4. Invoke Service (CAPTCHA Validierung) – Der aufgefundene Dienst wird zusammen mit den Daten der Anfrage ausgeführt.
5. Return Result to Engine (CAPTCHA Validierung) – Das erhaltene Resultat der Ausführung wird zurück an SBPEL Execution Engine gesendet.
6. Achieve Goal — Return Result to Engine – Die Schritte 2 bis 4 wiederholen sich für jeden Prozessschritt mit den jeweilig verknüpften Zieldefinitionen.
7. Return Result – Nachdem jeder Prozessschritt abgearbeitet wurde, wird das Resultat an den Aufrufenden gesendet. Im Beispiel entspricht das Resultat, bei einer erfolgreichen Ausführung, dem Listing 6.6.

Listing 6.6: IP SUPER – Resultat des Prozesses

```
instance cResponse memberOf glb#CaptchaValidationResponse
    glb#result hasValue true
```

```
instance aResponse memberOf glb#AddressValidationResponse
    glb#result hasValue true

instance sResponse memberOf glb#SaveAddressResponse
    glb#result hasValue true
```

Zur Demonstration einer praktischen Ausführung eines Geschäftsprozesses wird ein Beispiel aus einer Präsentation[1] für den 6nd International Semantic Web Conference (ISWC 2007) verwendet.

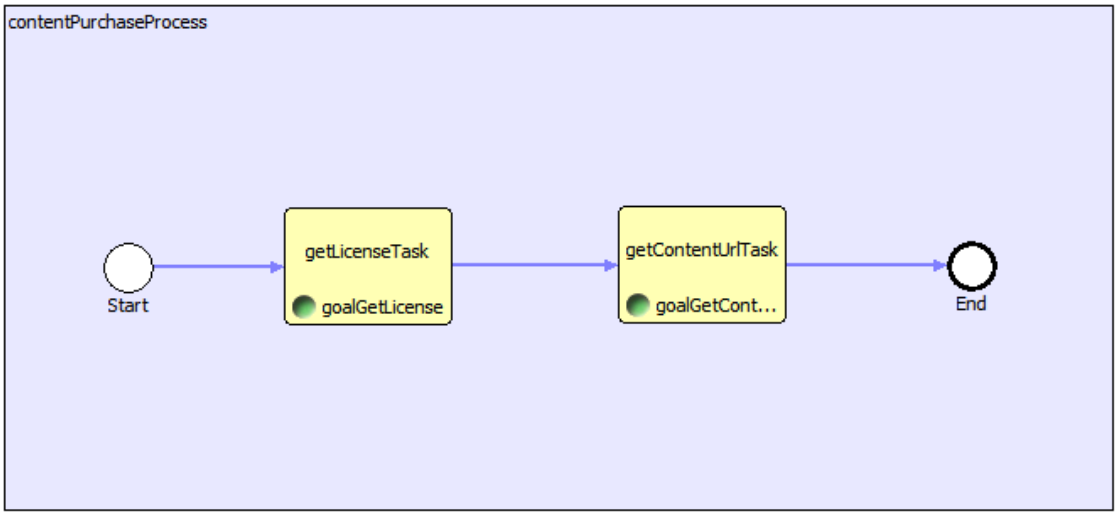

Abbildung 6.3: BPMO-Beispiel Prozess

Der Prozess in Abbildung 6.3 besitzt zwei Elemente, welche die Lizenzinformationen und die URL eines YouTube[2] Videos zurückliefern. Bei dem fertig modellierten Prozess handelt es sich um eine Prozessontologie 6.7, deren Instanzen von den Elementen der BPMO-Ontologie abgeleitet werden. In der BPMO-Ontologie sind die Beziehungen und Bedeutungen der Konzepte festgehalten, so dass eine Ausführung möglich ist.
Hauptelement der aufgeführten Ontologie ist die Prozess-Instanz, welcher eine Sequenz mit OrderElementen zugeordnet ist. Jedes OrderElement besitzt eine Nummer zur Bestimmung der Abarbeitungsreihenfolge „hasOrder“ und die Verknüpfung mit einem Prozess-Element.
Die GoalTasks enthalten einen Verweis auf das entsprechende Goal, welches während der

[1] http://www.ip-super.org/content/view/114/63/
[2] http://www.youtube.com

Ausführung einen Dienst ausfindig macht und dessen Endpunkt bestimmt, um die Anfrage an diesen weiterleiten zu können.

Listing 6.7: BPMO-Beispiel Ontologie

```
namespace { _"http://irs.open.ac.uk/contentPurchaseBpmoOntology#",
    bpmo _"http://irs.open.ac.uk/bpmo12#"
}

ontology contentPurchaseBpmoOntology

    importsOntology
        _"http://irs.open.ac.uk/bpmo12#bpmo12"

instance EndEvent_1 memberOf bpmo#EndEvent
    hasName hasValue "End"

instance StartEvent_1 memberOf bpmo#StartEvent
    hasName hasValue "Start"

instance GoalTask_1 memberOf bpmo#GoalTask
    hasName hasValue "getLicenseTask"
    hasWSMOGoal hasValue "http://irs.open.ac.uk/goalGetLicense#
        goalGetLicense"

instance Process_1 memberOf bpmo#Process
    hasName hasValue "contentPurchaseProcess"
    hasWorkflow hasValue bpmo#Sequence_1

instance Sequence_1 memberOf bpmo#Sequence
    hasOrderedElement hasValue {OrderedElement_1, OrderedElement_2,
        OrderedElement_3, OrderedElement_4 }

instance OrderedElement_1 memberOf bpmo#OrderedElement
    hasOrder hasValue 4
    hasElement hasValue EndEvent_1

instance OrderedElement_2 memberOf bpmo#OrderedElement
    hasOrder hasValue 3
    hasElement hasValue GoalTask_2

instance GoalTask_2 memberOf bpmo#GoalTask
```

```
36      hasName hasValue "getContentUrlTask"
        hasWSMOGoal hasValue "http://irs.open.ac.uk/goalGetContentUrl#
            goalGetContentUrl"
38
instance OrderedElement_3 memberOf bpmo#OrderedElement
40      hasOrder hasValue 1
        hasElement hasValue StartEvent_1
42
instance OrderedElement_4 memberOf bpmo#OrderedElement
44      hasOrder hasValue 2
        hasElement hasValue GoalTask_1
```

Zur Ausführung des Prozesses wird der IRS-III Server verwendet und als Repository in einem Eclipse Plug-In eingerichtet. Durch Auswahl des IRS-III Servers wird in Eclipse die Sicht auf die registrierten Elemente ermöglicht. Nach dem Import der BPMO-Ontologie wird automatisch, durch den IRS-III Server, ein Prozess Goal „process1Goal“ erzeugt. In der Abbildung 6.4 ist die automatisch erstellte Zieldefinition in der Auflistung der Goals zu erkennen.

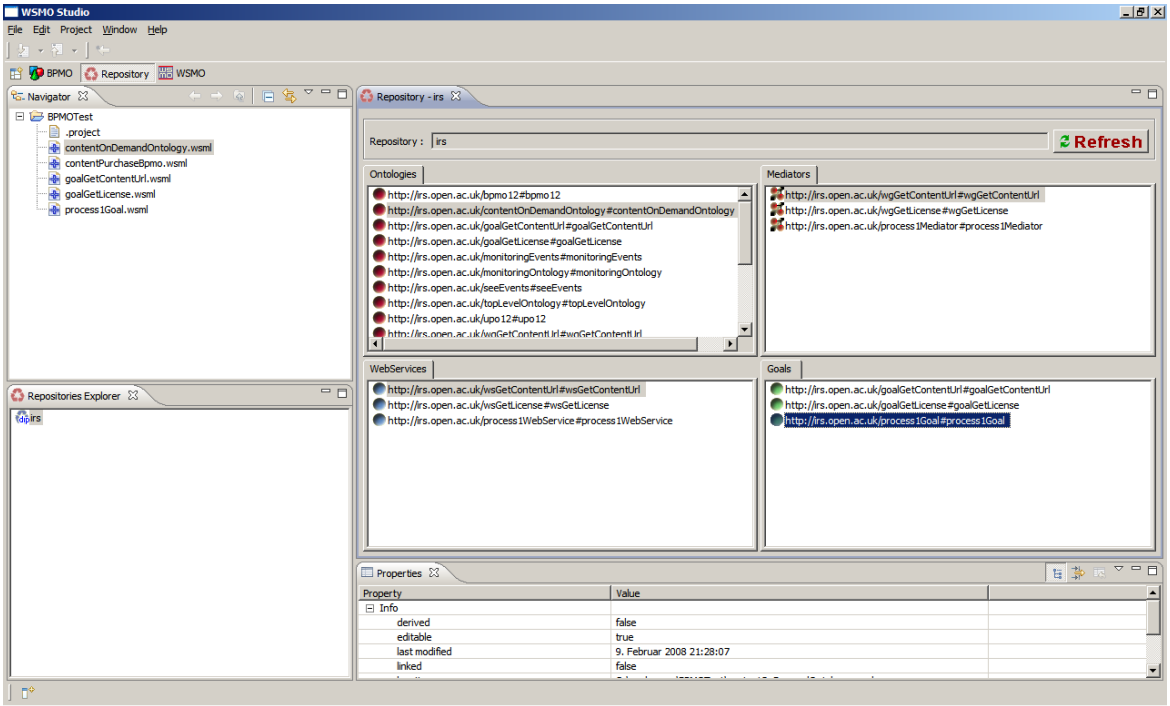

Abbildung 6.4: IRS-III Repository

Das generierte Prozess-Goal ist die Vereinigungsmenge der beiden Goals des Prozesses und kann genau wie eines der beiden verwendet werden. Durch Angabe einer Instanz mit

den Werten der Anfrage kann der Prozess ausgeführt werden. Im Beispiel wird die Instanz „simpleRequest"der ContentOnDemandOntology 6.8 verwendet.

Listing 6.8: Prozess Anfrage

```
ontology contentOnDemandOntology

concept reqGetLicense subConceptOf _"http://irs.open.ac.uk/
    baseOntology#ocmlThing"
     byUser ofType _string
     requestedContent ofType _string

concept resGetLicense subConceptOf _"http://irs.open.ac.uk/
    baseOntology#ocmlThing"
     license ofType _string

instance simpleRequest memberOf reqGetLicense
     byUser hasValue "carlos"
     requestedContent hasValue "football"
```

Das Resultat der Anfrage wird für jeden Schritt des Prozesses in einer für das Beispiel entwickelten Visualisierungssoftware angezeigt (Abbildung 6.5). In der ersten Spalte wird das Gesamtresultat des Prozesses dargestellt und in den beiden weiteren Spalten der Anwendung das Ergebnis der Teilabschnitte.

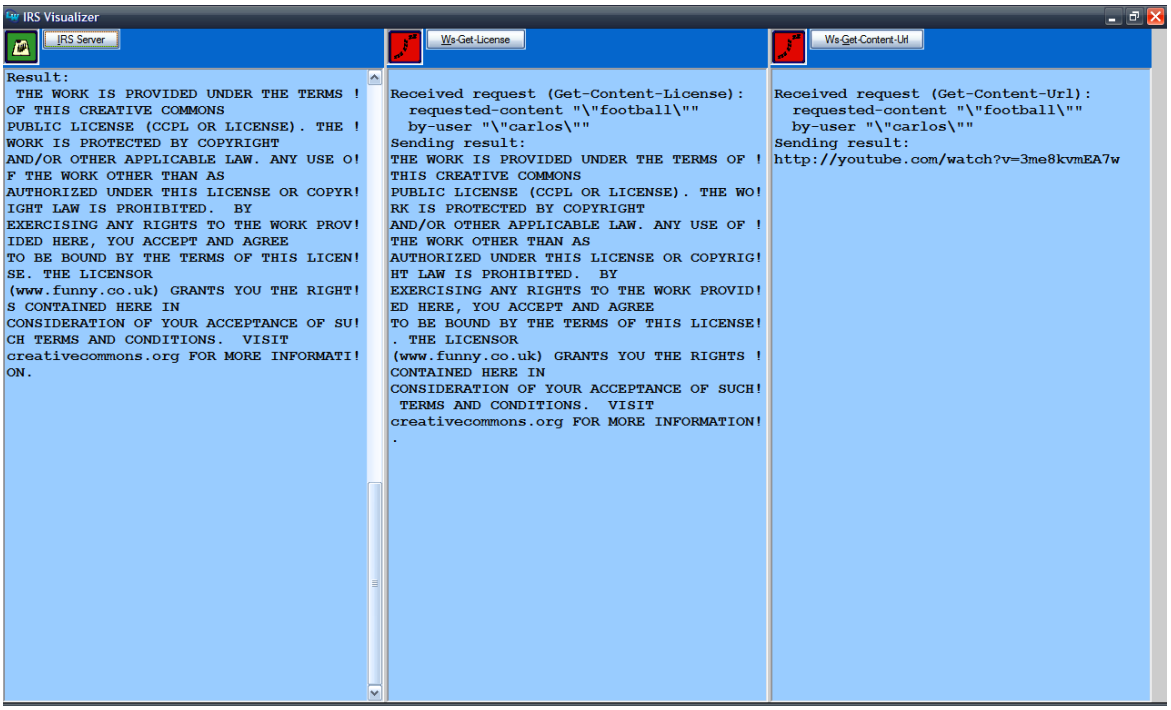

Abbildung 6.5: IRS Visualizer

7 Zusammenfassung und Ausblick

In der Arbeit wurden Technologien und Möglichkeiten aufgezeigt, einen Geschäftsprozess semantisch zu modellieren. Zur semantischen Beschreibung wurde die Web Service Modelling Ontology (WSMO) verwendet, da diese Entwicklung für die Verwendung in Geschäftsprozessen konzeptionell gut geeignet ist. Ein Grund dafür ist die konzeptionelle Trennung von Angebot und Anfrage. Konkret bezeichnet ein Angebot einen Web-Dienst und eine Anfrage für eine gewünschte Funktionalität. Im Zusammenhang mit einem Geschäftsprozess kann die gewünschte Funktionalität, welche bei WSMO als Goal bezeichnet wird, als Teilaufgabe des Prozesses verwendet werden.
Zur Ausführung eines semantischen Geschäftsprozesses wurde in der Arbeit die Entwicklung des IP SUPER Projektes vorgestellt. Zum jetzigen Zeitpunkt ist eine Ausführung mit dieser Entwicklung zwar nicht möglich, aber durch einen verfügbaren Beispielprozess wurde gezeigt, wie eine Ausführung durchgeführt wird.
Da im Moment an der Implementierung eines Prototypen zur Umsetzung des IP SUPER Konzepts gearbeitet wird, kann davon ausgegangen werden, dass eine Realisierung eines eigenen Geschäftsprozesses als Modellversuch bald möglich sein wird.
Anhand der veröffentlichten Informationen des IP SUPER Projekts lässt sich eine mögliche Richtung der Entwicklung absehen. Im Vergleich zu älteren Veröffentlichungen ist bei den neueren Konzeptvorstellungen keine vollständige Interpretion des semantischen Prozesses zur Laufzeit mehr vorgesehen. Es wird aktuell der Ansatz verfolgt, einen durch eine Ontologie beschriebenen Prozess in einen um Semantik erweiterte technische Beschreibung eines Prozesses zu transformieren. Die Transformation würde beim Anlegen eines Prozesses in der Execution Engine erfolgen und hätte den Vorteil, dass der Ablauf des Prozesses nicht während der Ausführung bestimmt werden müsste und somit die Abarbeitung schneller durchgeführt werden könnte. Weiterhin würde dieses Vorgehen eine höhere Prozesssicherheit gewährleisten, da durch eine technische Beschreibung der Ablauf festgelegt ist und dieser z.B. durch eine Änderung an einer Ontologie nicht beeinflusst werden würde.
Der Zeitpunkt, ab wann es möglich sein wird, ausschließlich semantische Geschäftsprozesse in der Praxis einzusetzen, ist nicht absehbar. Dennoch ist es aktuell vorstellbar, semantisch modellierte Prozesse als Basis für eine Transformation in einen technischen Prozess zu verwenden, um damit eine Erleichterung bei der Erstellung der Prozesse zu schaffen.

8 Anhang

Listing 8.1: Globale Ontologie

```
namespace { _"http://www.newsarea.de/wsmo/ontologies/global/v1.0/
    Global#"
,
      wsmostudio _"http://www.wsmostudio.org#",
      dc _"http://purl.org/dc/elements/1.1#",
      loc _"http://www.newsarea.de/wsmo/ontologies/location/v1.0/
          Location#" }

ontology _"http://www.newsarea.de/wsmo/ontologies/global/v1.0/
    Global.wsml"
          nonFunctionalProperties
          wsmostudio#version hasValue "0.7.2"
          endNonFunctionalProperties

          importsOntology {
          _"http://www.newsarea.de/wsmo/ontologies/location/v1.0/
              Location.wsml"
          }

concept ImageFormat
      name impliesType _string

concept Image
      width impliesType _integer
      height impliesType _integer
      format impliesType ImageFormat

concept Captcha subConceptOf Image
      id impliesType _string
      value impliesType _string
      length impliesType _integer
```

```
28
concept CaptchaValidationResponse
30         result impliesType _boolean

32 instance JPEG memberOf ImageFormat
       name hasValue "JPEG"
34
instance PNG memberOf ImageFormat
36       name hasValue "PNG"

38 instance BMP memberOf ImageFormat
       name hasValue "BMP"
40
/* ------------------------------
42                     ADDRESS
------------------------------ */
44
axiom cityByZIPandCountry
46         definedBy
                   ?address[
48                         zip hasValue ?zip,
                           loc#country hasValue ?country
50                 ] memberOf Address
                   and ?city[
52                         loc#zip hasValue ?zip,
                           loc#country hasValue ?country
54                 ] memberOf loc#City
                   implies ?address[
56                         city hasValue ?city
                   ].
58
axiom countryByCity
60         definedBy
                   ?address[
62                         city hasValue ?city
                   ] memberOf Address
64                 and ?city[loc#country hasValue ?country] memberOf
                       loc#City
                   implies ?address[
66                         country hasValue ?country
```

```
                ].

concept Address
        name impliesType _string
        company impliesType _string
        street impliesType _string
        streetNumber impliesType _string
        localityExtension impliesType _string
        building impliesType _string
        zip impliesType _string
        city impliesType loc#City
        country impliesType loc#Country

concept AddressValidationResponse
        result impliesType _boolean
        message impliesType _string

concept SaveAddressResponse
        result impliesType _boolean
```

Listing 8.2: Location Ontologie

```
namespace { _"http://www.newsarea.de/wsmo/ontologies/location/v1.0/Location#"
,
    wsmostudio _"http://www.wsmostudio.org#",
    dc _"http://purl.org/dc/elements/1.1#" }

ontology _"http://www.newsarea.de/wsmo/ontologies/location/v1.0/Location.wsml"
    nonFunctionalProperties
        wsmostudio#version hasValue "0.7.2"
    endNonFunctionalProperties

concept Country
    name impliesType _string

instance Germany memberOf Country
        name hasValue "Germany"

instance Greatbritain memberOf Country
        name hasValue "Greatbritain"

concept City
        name impliesType _string
        country impliesType Country
        zip impliesType _string

instance Hamburg memberOf City
        name hasValue "Hamburg"
        country hasValue Germany
        zip hasValue {"22087", "20097"}

instance London memberOf City
        name hasValue "London"
        country hasValue Greatbritain
        zip hasValue {"SW1X 8PZ", "NW5–2TJ"}
```

Listing 8.3: Communication Ontologie

```
namespace { _"http://www.newsarea.de/wsmo/ontologies/communication/
    v1.0/Communication#"
,
      wsmostudio _"http://www.wsmostudio.org#",
      dc _"http://purl.org/dc/elements/1.1#",
      loc _"http://www.newsarea.de/wsmo/ontologies/location/v1.0/
          Location#" }

ontology _"http://www.newsarea.de/wsmo/ontologies/communication/v1
    .0/Communication.wsml"
      nonFunctionalProperties
            wsmostudio#version hasValue "0.7.2"
      endNonFunctionalProperties

      importsOntology
          _"http://www.newsarea.de/wsmo/ontologies/location/v1.0/
              Location.wsml"

axiom CellularPhoneNetworkByPrefix
      definedBy
          ?phNum[
                  prefix hasValue ?prefix
          ] memberOf PhoneNumber
          and ?network[
                  prefix hasValue ?prefix
          ] memberOf CellularPhoneNetwork
                  implies ?phNum[
                          network hasValue ?network
                  ].

concept Transmission
      sender impliesType ContactInfo
      receiver impliesType ContactInfo
      subject impliesType _string
      body impliesType _string

concept ContactInfo

concept PhoneNumber subConceptOf ContactInfo
```

```
36      countryCode impliesType _string
        prefix impliesType _string
38      number impliesType _string
        extension impliesType _string
40      network impliesType PhoneNetwork

42  concept EMailAddress subConceptOf ContactInfo
        user impliesType _string
44      domain impliesType _string
        topLevelDomain impliesType _string
46
    concept PhoneNetwork
48      name impliesType _string
        country impliesType loc#Country
50
    concept CellularPhoneNetwork subConceptOf PhoneNetwork
52      prefix impliesType _string

54  instance O2_Germany memberOf CellularPhoneNetwork
        prefix hasValue {"0179", "0176" }
56      name hasValue "O2 (Germany) GmbH & Co. OHG"
        country hasValue loc#Belgium
58
    instance EPlus_Germany memberOf CellularPhoneNetwork
60      prefix hasValue { "0155", "0157", "0163", "0177", "0178" }
        name hasValue "E-Plus Service GmbH & Co.KG"
62      country hasValue loc#Germany
```

Literaturverzeichnis

[BPMN:2006 2006] : *BPMN Finalization Task Force to the Domain Technical Committee*. 2006. – URL http://www.omg.org/docs/dtc/06-11-01.pdf

[XML:2007 2007] : *Extensible Markup Language (XML)*. 2007. – URL http://www.w3.org/XML/

[Bechhofer u. a. 2004] BECHHOFER, Sean ; HARMELEN, Frank van ; HENDLER, Jim ; HORROCKS, Ian ; MCGUINNESS, Deborah L. ; PATEL-SCHNEIDER, Peter F. ; STEIN, Lynn A.: *OWL Web Ontology Language - Reference*. 2004. – URL http://www.w3.org/TR/owl-ref/

[Berners-Lee 1994] BERNERS-LEE, Tim: *A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web*. 1994. – URL http://tools.ietf.org/html/rfc1630

[Berners-Lee 1999] BERNERS-LEE, Tim: *Transcript of Tim Berners-Lee's talk to the LCS 35th Anniversary celebrations, Cambridge Massachusetts*. 1999. – URL http://www.w3.org/1999/04/13-tbl.html

[Berners-Lee u. a. 2007] BERNERS-LEE, Tim ; CONNOLLY, Dan ; HAWKE, Sandro ; HERMAN, Ivan ; PRUD'HOMMEAUX, Eric ; SWICK, Ralph: *W3C Semantic Web Activity*. 2007. – URL http://www.w3.org/2001/sw

[Bray u. a. 2006] BRAY, Tim ; PAOLI, Jean ; SPERBERG-MCQUEEN, C. M. ; YERGEAU, Eve Malerand F.: *Extensible Markup Language (XML) 1.0 (Fourth Edition) - W3C Recommendation 16 August 2006*. 2006. – URL http://www.w3.org/TR/xml/

[Brickley und Guha 2004] BRICKLEY, Dan ; GUHA, R.V.: *RDF Vocabulary Description Language 1.0: RDF Schema*. 2004. – URL http://www.w3.org/TR/rdf-schema/

[de Bruijn u. a. 2005] BRUIJN, Jos de ; BUSSLER, Christoph ; DOMINGUE, John ; FENSEL, Dieter ; HEPP, Martin ; KELLER, Uwe ; KIFER, Michael ; KÖNIG-RIES, Birgitta ; KOPECKY, Jacek ; LARA, Rubén ; LAUSEN, Holger ; OREN, Eyal ; POLLERES, Axel ; ROMAN, Dumitru ; SCICLUNA, James ; STOLLBERG, Michael: *Web Service Modeling Ontology (WSMO) - W3C Member Submission 3 June 2005*. 2005. – URL http://www.w3.org/Submission/WSMO/

[Bussler u. a. 2005] BUSSLER, Christoph ; MORAN, Matthew ; MURAN, Dumitru ; STOLLBERG, Michael ; ZAREMBA, Michael: *Semantic Web Services for Autonomic Computing: A Conceptual Model, Language, and Execution Environment*. 2005. – URL http://www.wsmo.org/TR/d17/resources/200506-ICAC/icac-05_tutorial.ppt

[Christensen u. a. 2001] CHRISTENSEN, Erik ; CURBER, Francisco ; MEREDITH, Greg ; WEERAWARANA, Sanjiva: *Web Services Description Language (WSDL) 1.1*. 2001. – URL http://www.w3.org/TR/wsdl

[Duerst und Suignard 2005] DUERST, M. ; SUIGNARD, M.: *Internationalized Resource Identifiers*. 2005. – URL http://tools.ietf.org/html/rfc3987

[Eisler 1904] EISLER, Rudolf: *Wörterbuch der philosophischen Begriffe*. Herausgeber Unbekannt, 1904. – URL http://www.textlog.de/4761.html

[Fensel u. a. 2007] FENSEL, Dieter ; LAUSEN, Holger ; POLLERES, Axel ; STOLLBERG, Michael ; ROMAN, Dumitru ; BRUIJN, Jos de ; DOMINGUE, John: *Enabling Semantic Web Services*. Springer-Verlag Berlin Heidelberg, 2007

[Gruber 1993] GRUBER, Thomas R.: *A Translation Approach to Portable Ontology Specifications - Abstract*. 1993. – URL http://ksl-web.stanford.edu/KSL_Abstracts/KSL-92-71.html

[Gudgin u. a. 2007] GUDGIN, Martin ; HADLEY, Marc ; MENDELSOHN, Noah ; MOREAU, Jean-Jacques ; NIELSEN, Henrik F. ; KARMARKAR, Anish ; LAFON, Yves: *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. 2007. – URL http://www.w3.org/TR/soap12-part1/

[Herman 2007] HERMAN, Ivan: *Introduction to the Semantic Web*. 2007. – URL http://www.w3.org/2007/Talks/0831-Singapore-IH/Slides.pdf

[Koivunen und Miller 2001] KOIVUNEN, Marja-Riitta ; MILLER, Eric: *W3C Semantic Web Activity*. 2001. – URL http://www.w3.org/2001/12/semweb-fin/w3csw

[Krafzig u. a. 2007] KRAFZIG, Dirk ; BANKE, Karl ; SLAMA, Dirk: *Enterprise SOA*. Mitp-Verlag, 2007

[Lara u. a. 2005] LARA, Ruben ; POLLERES, Axel ; LAUSEN, Holger ; ROMAN, Dumitru ; BRUIJN, Jos de ; FENSEL, Dieter: *A Conceptual Comparison between WSMO and OWL-S*. 2005. – URL http://www.wsmo.org/TR/d4/d4.1/v0.1/d4.1v0.1_20050412.pdf

[Linck 2006] LINCK, Christoph: *Semantic Web Services*. 2006. – URL http://www.st.informatik.tu-darmstadt.de/database/seminars/data/linck.pdf?id=206

[Manola und Miller 2004] MANOLA, Frank ; MILLER, Eric: *RDF Primer*. 2004. – URL http://www.w3.org/TR/rdf-primer/

[McGuinness und van Harmelen 2004] MCGUINNESS, Deborah L. ; HARMELEN, Frank van: *OWL Web Ontology Language - Overview*. 2004. – URL http://www.w3.org/TR/owl-features/

[Melze 2007] MELZE, Ingo: *Service-orientierte Architekturen mit Web Services*. Spektrum Akademischer Verlag; 2. Auflage, 2007. – URL http://www.soa-buch.de

[Nelson 1974] NELSON, Theodor H.: *Computer Lib / Dream Machines*. The Distributors, 1974

[Pellegrini und Blumauer 2006] PELLEGRINI, Tassilo ; BLUMAUER, Andreas: *Semantic Web*. Springer-Verlag Berlin Heidelberg, 2006

[Roman u. a. 2007] ROMAN, Dumitru ; LAUSEN, Holger ; KELLER, Uwe: *D2v1.4. Web Service Modeling Ontology (WSMO)*. 2007. – URL http://www.wsmo.org/TR/d2/v1.4/

[SUPER 2007] SUPER, IP: *Integrated Project SUPER - Showcase*. 2007. – URL http://www.ip-super.org/res/070928_D10.2_Showcase.pps

[Vömchen 2005] VÖMCHEN, Wikipedia-User: *Bild - Funktion Webservice*. 2005. – URL http://de.wikipedia.org/wiki/Bild:Webservice.png

Abkürzungsverzeichnis

BPEL	Business Process Execution Language
BPMN	Business Process Modeling Notation
BPMO	Business Process Management Ontology
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
DAML	The DARPA Agent Markup Language
DARPA	The Defense Advanced Research Projects Agency
ESSI	European Semantic Systems Initiative
F-Logik	Frame Logik
IETF	The Internet Engineering Task Force
IP SUPER	Integrated Project SUPER (Semantics Utilised for Process Management within and between Enterprises)
IRI	Internationalized Resource Identifier
IRS	Internet Reasoning Service
OIL	Ontology Inference Layer
OMG	Object Management Group
OWL	Web Ontologie Language
OWL-S	Web Ontology Language for Web Services
RDF	Resource Description Framework
RDF-S	Resource Description Framework Schema
RIF	Rule Interchange Format
SBPELEE	Semantic BPEL Execution Engine
SEE	Semantic Execution Environment
SGML	Standard Generalized Markup Language
SOA	Serviceorientierte Architektur
SPARQL	SPARQL Protocol and RDF Query Language
URI	Uniform Resource Identifier
W3C	The World Wide Web Consortium

WSDL ..... Web Services Description Language
WSML ..... Web Service Modelling Language
WSMO ..... Web Service Modelling Ontology
WSMX ..... Web Service Modelling Execution Environment
XML ..... Extensible Markup Language

Tabellenverzeichnis

Abbildungsverzeichnis

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 27. Februar 2008	
Ort, Datum	Unterschrift