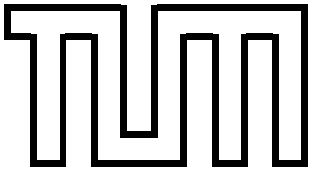
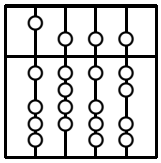


Technische Universität
München
Fakultät für Informatik



Forschungs- und Lehreinheit Informatik III Prof. R. Bayer Ph.D.,
Prof. Dr. D. Kossmann Hauptseminar Webservices

Sommersemester 2003

Web und Ontologien

Ausarbeitung

Referent: Otmar Hilliges
Betreuer: Wolfgang Wohner
Abgabedatum: 08.05.2003
Vortragsdatum: 15.05.2003

Contents

Chapter 1

Einführung

Das Internet hat sich in den letzten Jahren zu einem Massenmedium entwickelt und ist zu einer sehr großen Quelle für Informationen aller Art angewachsen. Im Jahr 2000 wurde die Benutzerzahl auf 350 Millionen geschätzt [Waller]. Diese Entwicklung und damit die Flut der Informationen hat immer mehr an Fahrt gewonnen. Die Entwicklung wird sich auch in Zukunft fortsetzen. Schon heute ist es, trotz Suchmaschinen, nicht immer einfach, die benötigten Informationen in angemessener Zeit zu finden. Das liegt vor allem daran, dass das World Wide Web und seine Beschreibungssprachen (HTML) dafür ausgelegt sind, Informationen für Menschen lesbar aufzubereiten.
In den frühen Tagen des Web wurden Webseiten hauptsächlich von Hand erstellt und enthielten außer dem eigentlichen Text und Formatierungsanweisungen (Tags) keinerlei Hinweise auf Struktur, Art und Semantik der dargestellten Information. In der nächsten Phase bereiteten Web-Editoren aller Art und ein damit einhergehender Qualitätsanstieg der Webseiten den Weg in unser tägliches Leben und somit auch in alle Sparten der Wirtschaft. Das Hauptaugenmerk blieb trotzdem auf der Lesbarkeit für Menschen und nicht für Maschinen. Trotz weit entwickelter Katalogiesierungs- und Rankingmechanismen wie sie Suchmaschienen wie z.B. Google [Google, Franz Embacher] verwenden, müssen zum Beurteilen der Resultate und zuletzt zum Filtern der Informationen immer noch Menschen eingreifen, um die gewünschten Antworten auf ihre Fragen zu erhalten. Deutlich wird das, wenn man bei Google den 'Auf gut Glück' Button verwendet, welcher den Benutzer automatisch auf den ersten Treffer weiterleitet. Diese Seiten enthalten in nicht wenigen Fällen hauptsächlich irrelevante Informationen.

Um die Masse der verfügbaren Informationen und Dienste effektiver zu nutzen, sollte das Web maschinenfreundlicher werden. Als Konsequenz folgt aus dieser Erkenntnis, dass bei der Entwicklung (und der Nutzung) von neuen Webinhalten ein Schwerpunkt auf Maschinenlesbarkeit und Verarbeitbarkeit von Informationen gelegt werden sollte. Das kann dadurch geschehen, dass Webseiten und WebServices mit Meta-Informationen angereichert werden, die es Maschinen erlauben, die richtigen Ergebnisse auf eine Suchanfrage zu liefern. Die Grundlagen hierzu liefern Technologien wie XML[TimBray et al., 2000] und RDF[Lassila, 1998]. Wenn einmal möglichst viele Webseiten mit (maschinenlesbaren) Hin-

weisen über die Art und Struktur sowie Semantik der enthaltenen Informationen versehen sind, wird es für Suchmaschinen und künstliche Agenten[1] um einiges einfacher, Informationen zu finden, zu bewerten und evtl. sogar aufzubereiten.
Eine Ebene höher sind dann die hier zu besprechenden Ontologien anzusiedeln. Ontologien werden verwendet, um eine Wissensdomäne (wie z.B. Biologie, Mechanik, Medizin etc.) und die Beziehungen ihrer Inhalte untereinander zu beschreiben und zu katalogisieren. Zum Beispiel ist *WordNet*[2] [Fellbaum, 1999] ein Online lexikalisches Referenz-System, das auf neuesten Erkenntnissen aus der psycholinguistischen Forschung über das menschlich-lexikalische Gedächtnis aufbaut. Englische Nomen, Verben, Adjektive und Adverbien sind in Synonymen Mengen organisiert, wobei jede Menge jeweils ein lexikalisches Konzept (eine 'Bedeutung') repräsentiert. Die einzelnen Wörter-Gruppen werden untereinander verknüpft, indem ihre Beziehungen zu anderen Gruppen modelliert werden mit Beziehungen wie:

Synonym zwei Konzepte bedeuten dasselbe.

Dichotemie Einteilung des Begriffsumfanges in zwei zueinander komplementäre Artbegriffe. Ein Begriff A wird in zwei Begriffe B und Nicht-B eingeteilt, die den Umfang des Begriffes A vollständig umfassen.

Hyponomie Eine 'is-a' Beziehung zwischen Konzepten. Sie wird verwendet, um Hierarchien zu modellieren und die Eigenschaftsvererbung von 'Superkonzepten' zu 'Subkonzepten' darzustellen.

WordNet ist ist freizugänglich und kostenlos, durch seine Ausrichtung auf Konzepte ist es viel aussagekräftiger als ein einfaches Lexikon.

1.1 Ontologien

Um Ontologien und ihre Anwendung genauer betrachten zu können, muss man klären woher der Begriff kommt, was er im Kontext der Informatik bedeutet und wie der Begriff zu verstehen und anzuwenden ist.

1.1.1 Ursprung des Begriffs

Der Begriff Ontologie kommt Ursprünglich aus der Philosophie und taucht dort seit Mitte des 18ten Jahrhunderts auf. In der Philosophie ist der Begriff sehr eng mit dem der Metaphysik verwandt und wird teilweise als Synonym verwendet. Ontologie ist also die Wissenschaft, die sich mit dem Sein und der Existenz im allgemeinen beschäftigt, im Internet findet sich unter [Principia Cybernetica Web] folgende Definition:

> Ontologie untersucht die Frage, warum die Dinge so sind, wie sie sind

[1] Autonom agierende Programme, die mit künstlicher Intelligenz ausgestattet sind.
[2] http://www.cogsci.princeton.edu/ wn/

1.1.2 Definition

In letzter Zeit wurde der Begriff von der Informatik übernommen. Zuerst wurde er im Zusammenhang mit Künstlicher Intelligenz (KI) verwendet. In der KI werden mit Ontologien Modelle unserer Welt bezeichnet, die verwendet werden, um (künstlichen) Agenten die semantischen Zusammenhänge zwischen Elementen dieser Welt zu beschreiben. Es handelt sich also im Grunde um ein Vokabular, um verteiltes Wissen zu beschreiben und gemeinsam zu nutzen sowie um die Elemente eines Wissensbereiches eindeutig zu benennen und die Beziehungen zu den anderen Elementen zu definieren. Ausserdem wird dieses Vokabular verwendet, um Suchanfragen und Antworten eindeutig zu gestalten. Ontologien garantieren durch ihr eindeutiges Vokabular Konsistenz aber keine Vollständigkeit. Deswegen muss, um Wissen zu teilen, nicht die gesamte Informationsmasse verteilt werden, sondern nur die aktuell gewünschte Information transferiert werden.
Laut Tom Gruber [Gruber] handelt es sich also um die

> Spezifikation einer Konzeptualisierung,

wobei unter einer Konzeptualisierung eine abstrakte, vereinfachte und formalisierte Ansicht der Welt, oder Wissensdomäne zu verstehen ist, die wir repräsentieren wollen. Dabei bezieht sich formalisiert auf die Maschinenlesbarkeit.
Ontologien beinhalten also auch taxonomische Hierarchien, um Klassen von Objekten und ihre Beziehungen untereinander zu beschreiben, allerdings bieten Ontologien darüberhinaus die Möglichkeit, den so eingeführten Begriffen eine Erklärung über deren Bedeutung zur Seite zu stellen.

Zusammengefaßt spezifizieren Ontologien eine detaillierte Beschreibung der

- Terminologie
- Konzepte
- Beziehungen zwischen den Konzepten
- Regeln

die für eine bestimmte Wissensdomäne gelten.

Ontologien in diesem Sinne finden ihre Anwendung in vielen Bereichen der Informatik, darunter Disziplinen wie knowledge engineering, knowledge representation, qualitative modeling, language engineering, database design, information retrieval and extraction, and knowledge management and organization und data mining. Speziell im Internet finden Ontologien ihre Anwendung in Bereichen wie ontology-enhanced search, library-science und dem im Zusammenhang mit WebServices besonders interessanten eCommerce.

1.1.3 Typen von Ontologien

Ontologien werden in vielen unterschiedlichen Bereichen und meist von einer größeren Gruppe, die über die ganze Welt verteilt sein kann, entwickelt und verwendet. Das führt dazu, dass Ontologien sich in Aspekten wie Umfang, Allgemeingültigkeit vs. Spezialisierung, Granularität oder dem Grad der Formalisierung unterscheiden. Trotzdem lassen sich die meisten Ontologien unter anderem in folgende Klassen einteilen [D.Fensel, 2000],[Dutra, 2001]:

Upper-Level Ontologien bilden die Grundlage, auf denen andere Ontologien erstellt werden.

Generische Ontologien beinhalten Informationen über allgemeingültiges Wissen über die Welt und liefern damit vereinfachtes Wissen über Zeit und Raum etc. Generische Ontologien sind naturgemäß in mehreren *Domänen Ontologien* gültig.

Domänen Ontologien erfassen das Wissen einer speziellen Wissensdomäne (z.B.: Medizin, Biologie, Mechanik etc.)

Metadaten Ontologien liefern ein Vokabular, um Online Informationen und deren Bedeutung zu beschreiben (z.B.: Dublin Core [Weibel et al., 1995]).

Interface Ontologien definieren Struktur und Inhalt sowie deren Grenzen und die Anwendung eines bestimmten Interfaces (z.B.: einer bestimmten API).

Rollen Ontologien definieren die Terminologie und Konzepte, die für eine bestimmte Rolle (Benutzer, Applikation) gelten.

Representational Ontologies gehören nicht zu einer speziellen Wissensdomäne. Sie beschreiben lediglich Container, ohne Aussagen darüber zu treffen, was für Information diese Container beinhalten. Solche Ontologien erlauben es, Information objektorientiert darzustellen.

Um den Nutzen von Ontologien auszuschöpfen, sollten diese wiederverwendbar sein. Um dieses Ziel, zu erreichen müssen Ontologien als möglichst kleine Module mit starken Verknüpfungen innerhalb des Moduls sowie schwachen und wenigen Verknüpfungen nach aussen entwickelt werden. Mit nach obigen Richtlinien und Klassifizierung entwickelten Ontologien ließen sich auf einfache Weise neue Ontologien erstellen, die dann einen viel größeren oder eventuell anders gearteten Problembereich abdecken können.

Chapter 2

SemanticWeb und Ontologien

Um die Anwendungsbereiche von Ontologien im Bereich WebServices zu betrachten, muss man auch das SemanticWeb betrachten, denn in Entwickler- und Anwender-Kreisen schwelt zur Zeit ein Streit [Clark, 2003] über die Zukunft des Internets. Die eine Seite (vorwiegend Akademiker) sieht diese im SemanticWeb, die andere (hauptsächlich Firmen) in WebServices. Die Frage ist, ob dieser Streit berechtigt ist oder ob sich nicht beide Begriffe überlappen oder sogar synonym sind. SemanticWeb bedeutet, statischen Inhalt von Webseiten, die wie oben beschrieben für Menschen entwickelt wurden, für Maschinen lesbar und interpretierbar zu machen. Um das Konzept des SemanticWeb zu realisieren, gibt es bereits eine Reihe von Technologien. Die wichtigsten sind XML, RDF und Ontologien.
WebServices sind Dienste, die es Applikationen ermöglichen, über das Internet zu kommunizieren und Daten auszutauschen. Sie finden ihre Anwendung vor allem im eCommerce. Sowohl für SemanticWeb als auch für WebServices gibt es eine Reihe von Standards des *World Wide Web Consortiums* (http://w3.org). Die Kombination von SemanticWeb, und im speziellen Ontologien, sowie WebServices erlaubt es, Intelligente WebServices zu entwickeln, deren Nutzen über den simplen Datenaustausch hinaus geht. Bevor näher auf die einzelnen Aspekte eingegangen wird, soll die Graphik 2 die Zusammenhänge verdeutlichen. Aus meiner Sicht ist oben erwähnter Streit also eher ein Streit über Taxonomien als über Inhalte. Verdeutlicht wird diese Annahme durch die Verwebung beider Begriffe in Neuen Technologien wie DAML-S, mehr dazu jedoch in Kapitel 4

2.1 SemanticWeb - die nächste Stufe des Internets

Das SemanticWeb ist kein neues Web im eigentlichen Sinne, sondern eine Erweiterung des bestehenden WorlWideWebs. Wie das WWW ist auch das SemanticWeb dezentral und muss somit entsprechende eingehen in Bezug auf Konsistenz und Vollständigkeit, was in beiden Fällen nicht erreicht oder gar garantiert werden kann.

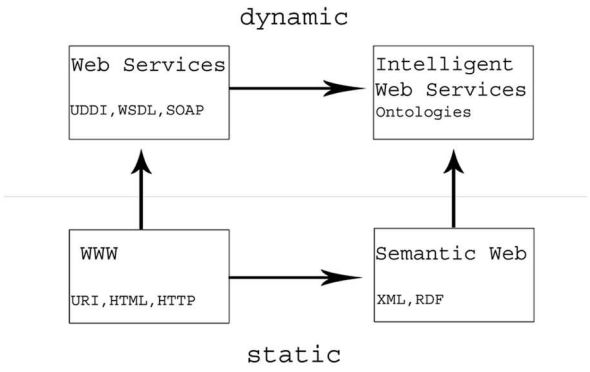


Figure 2.1: Beziehungen zw. SemanticWeb, WebServices und Ontologien.

2.1.1 Was ist neu am SemanticWeb?

Der Hauptunterschied zum herkömmlichen WWW liegt darin, dass wichtige Informationen wohl definiert strukturiert werden und mit 'Aussagekraft' für Maschinen und Programme versehen werden, so dass zeitaufwendige und lästige Aufgaben im Internet von Maschinen übernommen werden können. Dazu gehören Aufgaben wie Informationen finden und strukturieren, Informationen und Wissen mit anderen austauschen, Terminpläne abstimmen und vieles mehr, was heutzutage noch gar nicht absehbar ist.

2.1.2 Ein Beispiel: der Intelligente Terminplan

Folgendes Szenario verdeutlicht mögliche Anwendungen für das SemanticWeb. Ein Agent, ausgestattet mit Künstlicher Intelligenz, könnte mit Hilfe des SemanticWeb einen Arzt, in der näheren Umgebung eines Patienten finden, der auf eine spezielle Krankheit spezialisiert ist. Der Agent könnte gleichzeitig Termine für eine oder mehrere Behandlungssitzungen mit dem ebenfalls elektronischen Terminplan des ausgewählten Arztes vereinbaren. Weiter könnte der Agent Verkehrs-Verbindungen vom Wohnsitz des Patienten zur Praxis heraussuchen. Und all das ganz ohne das Eingreifen des Patienten oder Arztes.

2.1.3 Technologien zur Realisierung

Ermöglicht würde dieses Szenario, indem die Webseiten der involvierten Personen und Institutionen, zusätzlich zum eigentlichen Text, durch XML Tags mit Informationen angereichert werden. So könnte auf der Seite des Arztes dessen Fachgebiet, die Adresse und die Sprechstunden sowohl als Text wie auch als maschinenlesbare Information, eingewoben in die zugrundeliegenden XML Tags, aufgeführt sein. Das Problem dabei ist, dass die XML Tags zwar Struktur in die Informationen bringen, aber für den Agenten aus unserem

Beispiel nichts bedeuten. Diese Hürde kann mit RDF genommen werden. RDF kodiert Semantik in einem Tripel, welches wiederum in XML-Tags beschrieben wird. Das Tripel ist Analog zu einem Satz aus Subjekt, Objekt und Prädikat zu verstehen, wobei das Prädikat die Relation von Subjekt und Objekt beschreibt.
RDF geht davon aus, dass beliebige Dinge (z.B: Personen, Dokumente oder Gegenstände) eine Eigenschaft haben, die ihre Beziehung zu anderen Dingen ausdrückt, wie 'Tom ist der Autor von Tom's Tagebuch' oder 'Anna ist die Tochter von Tom'. In RDF sind 'is-a' sowie 'is-subclass-of' Beziehungen standartisiert. Subjekt, Objekt und Prädikat werden jeweils von URIs (Universal Resource Identifier[1]) beschrieben, die wie normale Weblinks auf eine Beschreibung des Begriffs deuten. Dieser Ansatz reicht aus, um die meisten Daten zu beschreiben, die von Maschinen verarbeitet werden.
Ein Problem bleibt jedoch bestehen, denn ein Begriff kann mehrere Bedeutungen haben, und Maschinen haben nicht die Möglichkeit aus dem Kontext zu schließen, welche Bedeutung gemeint ist.

2.1.4 Ontologien

Dieses Problem kann zumindest teilweise durch die Zusammenfassung einzelner RDF Statements zu einer Ontologie gelöst werden. Denn Ontologien erlauben es, zusätzlich zu dem in RDF standardisierten objektorientierten Ansatz Äquivalenzrelationen zu definieren. So könnte in einer Ontologie der Begriff 'Adresse' als Privatadresse definiert sein und in einer anderen sowohl als Privatadresse sowie als Postfach. Wenn nun in einer der beiden Ontologien eine Relation definiert ist, die es einem Agenten erlaubt, das Postfach von einer 'echten' Adresse zu unterscheiden, wird der Patient aus obigem Beispiel nicht vor das Postfach einer Spezialklinik, sondern in deren Warteraum navigiert.
Darüber hinaus haben Ontologien natürlich noch mehr Vorteile, denn Ontologien beschreiben meist ein gesamtes Konzept. Also ließen sich zum Beispiel aus einer Ontologie, die von der persönlichen Seite eines Forschers verlinkt ist, Informationen herausfiltern, für die ein Mensch vermutlich über mehrere im Web verteilte Seiten browsen müsste. Angenommen es handelt sich bei dem Forscher um einen Professor, der an einer bestimmten Universität arbeitet. Da Professoren immer einen Doktor Titel tragen, könnte auch gleich die Universität, an der er promoviert hat, sowie die Forschungsgruppen in denen er involviert ist herausgefunden werden, auch e-mail Adresse und Raumnummer könnten in Erfahrung gebracht werden und letztendlich all diese Information gegliedert als Antwort auf eine komplexe Frage, die eben nicht an einer einzigen Stelle im Web zu beantworten ist, zurückgegeben werden.

[1] http://www.w3.org/Addressing/

Chapter 3

Sprachen und Technologien

Um oben beschriebene Anwendungen zu implementieren, sind natürlich gewisse Technologien und Sprachen Voraussetzung. Diese Technologien wurden teilweise schon kurz eingeführt, sollen jetzt aber genauer und betrachtet und erläutert werden. Grundkenntnisse in XML sind Voraussetzung.

3.1 RDF

XML definiert die Struktur eines Dokumentes, dabei wird ein Baum aufgespannt, und alle Blätter dieses Baumes sind wohldefinierte Tags. XML sorgt dafür, dass Dokumente von Maschinen gelesen werden können und macht den Austausch von Daten zwischen Maschinen möglich. Allerdings ermöglicht es XML Maschinen nicht, die Informationen, die in dem Dokument enthalten sind, zu verstehen. Dafür gibt es das Resource description framework (RDF) [Lassila & Swick, 1999]. RDF liefert Methoden und Konzepte, um Dokumenten Semantik hinzuzufügen, ohne Annahmen über deren Struktur zu machen. RDF adressiert auch keine Problemstellungen, die das Speichern, Transportieren usw. von Dateien angehen, sondern stützt sich in diesen Punkten vollkommen auf XML. RDF ist eine XML Anwendung, das bedeutet, dass die Syntax von RDF in XML beschrieben ist. RDF wurde entwickelt, um Meta-Informationen zu Web Dokumenten hinzuzufügen und um Wissensdomänen zu beschreiben. Dabei sind die Mechanismen von RDF so gestaltet, dass a priori keine direkte Verbindung oder Abhängigkeit zur beschriebenen Wissensdomäne besteht. Die RDF Mechanismen sind also Domänen neutral und deswegen geeignet jede beliebige Wissensdomäne zu beschreiben. Suchmaschinen, elektronische Agenten, Informationsbroker, Browser und letztendlich auch Menschen können Nutzen aus den semantischen Informationen ziehen. RDF ist ein W3C Standard und wird von anderen Standards wie PICS-2, P3P, und DigSig genutzt.

3.1.1 Das RDF Modell

Die Grundlage von RDF ist eine Modell, um bestimmte Eigenschaften und deren Werte zu beschreiben. RDF Eigenschaften kann man als Attribute bestimmter Ressourcen ansehen und eine in RDF beschriebene Ressource somit als Kombination aus Attribut-Wert Paaren. Das RDF Daten Modell läßt sich auch verwenden, um Beziehungen zwischen Ressourcen zu modellieren, und kann klassische Entity-Relationship Diagramme nachbilden. Übertragen auf Objekt-orientierte Terminologie entsprechen Ressourcen Klassen und Eigenschaften Instanz Variablen.
Das RDF Daten Modell beinhaltet drei Objekttypen: Subjekte, Prädikate und Objekte [Brickley, Guha 2003].

Subjekte sind Entitäten, die von einer URI (z.B. eine ganze Webseite oder ein spezielles XML Tag) beschrieben werden können.

Prädikate definieren eine binäre Relation zwischen Ressourcen und zwischen atomaren Werten von primitiven Datentypen, die in XML spezifiziert sind. Subjekte sind die Ressourcen, die in RDF Ausdrücken beschrieben werden.

Objekte spezifizieren einen Wert für ein bestimmtes Prädikat. Daraus ergibt sich, dass Objekte die eigentliche Charakterisierung von Dokumenten darstellen.

Beispiele

Als einfaches Beispiel [Lassila & Swick, 1999] wird der Satz

> 'Ora Lassila ist der Autor der Resource http://www.w3.org/Home/Lassila.'

betrachtet. Dieser beinhaltet folgende Satzteile:

Subjekt (Ressource)	http://www.w3.org/Home/Lassila
Prädikate (Eigenschaft)	Autor
Objekt (Literal)	'Ora Lassila'

Dieses Statement wird in RDF folgendermaßen ausgedrückt:

```
<rdf:RDF>
 <rdf:Description about="http://www.w3.org/Home/Lassila">
   <s:Creator>Ora Lassila</s:Creator>
 </rdf:Description>
</rdf:RDF>
```

Obige RDF-Tags sind etwas kryptisch zu lesen, deshalb kann man sich deren Bedeutung an dem aufgespannten Graphen (siehe Figure 3.1.1) verdeutlichen, dieser Graph ist semantisch identisch zu den Tags.

In RDF würde dieser Satz alleinstehend als Faktum interpretiert werden, allerdings kann man in RDF auch Statements über Statements formulieren. Zum Beispiel:

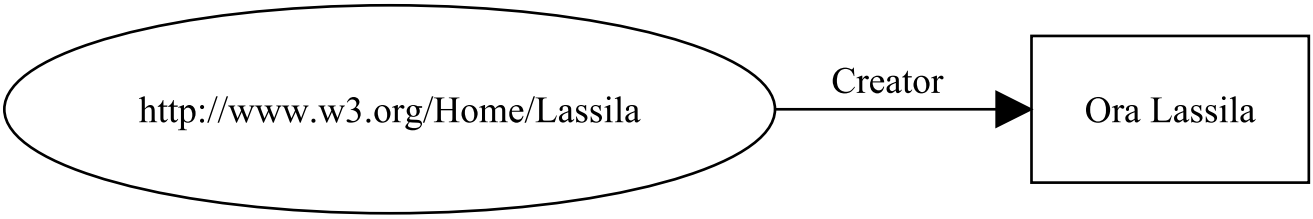


Figure 3.1: RDF Datenmodell.

'Ralph Swick behauptet, dass Ora Lassila Autor der Ressource http://www.w3.org/Home/Lassila ist'

In RDF kann man nun sowohl ausdrücken, dass Ralph Swick dieses Statement abgegeben hat, als auch die eigentliche Ressource beschreiben. Das nennt man ein 'reified Statement' (frei übersetzt: ein verifiziertes Zitat). Die RDF Syntax dieses reified Statements lautet:

```
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:a="http://description.org/schema/">
 <rdf:Description>
   <rdf:subject resource="http://www.w3.org/Home/Lassila" />
   <rdf:predicate resource="http://description.org/schema/Creator" />
   <rdf:object>Ora Lassila</rdf:object>
   <rdf:type resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement" />
   <a:attributedTo>Ralph Swick</a:attributedTo>
 </rdf:Description>
</rdf:RDF>
```

Die RDF-Syntax ist vom *W3C* standardisiert und im Internet [Lassila & Swick, 1999] nachzulesen. Zwei Aspekte von RDF sollten zum besseren Verständnis noch klar sein.

1. Das RDF Datenmodell ist Syntax neutral, bei der Äquivalenzprüfung zweier RDF-Statements kommt es also darauf an, ob ihre Datenmodelle identisch sind. Also ob die beiden aufgespannten Graphen (z.B: Figure 3.1.1) die selbe Bedeutung repräsentieren.

2. Obwohl RDF eine Äquivalenzprüfung ermöglicht, bietet es keine Mechanismen zur automatischen Beweisführung, wie sie aus Bereichen der *knowledge representation* und *description logic* bekannt sind. Dies wird jedoch von dem auf RDF aufbauenden DAML+OIL in Ansätzen ermöglicht.

3.1.2 RDFS

RDF beinhaltet auch Metadaten-Schemata [Brickley, Guha 2003] (RDFS). Ein Schema definiert die Bedeutung, die Eigenschaften und die Verhältnisse einer Menge von Ressourcen. RDF Schemata können mögliche Werte von Eigenschaften begrenzen und/oder bestimmte Eigenschaften von anderen Schemata erben bzw. an andere Schemata vererben. RDFS basiert auf dem gleichen Modell wie die RDF Syntaxspezifikation und fügt ihr 'Objekt orientierte' Konzepte wie Vererbung hinzu. Es gibt drei 'Super'-Klassen in RDFS; diese sind wie folgt definiert:

Ressource ist im Allgemeinen die Superklasse für alle Subjekte.

Property Type ist die Superklasse aller Prädikate

Class ist die Superklasse aller Werte von Prädikaten (Objekte)

Diese Drei Superklassen können durch folgende Eigenschaften miteinander verknüpft werden:

instanceOf definiert die Beziehung zwischen einer Ressource und einer Klasse.

subClassOf definiert die Beziehung zwischen zwei Elementen einer Klasse und ist transitiv.

Constraint ist eine *subClassOf PropertyType*. Ein *Constraint* hat zwei Basis Instanzen und zwar *range* und *domain*, die auf alle *Property Types* angewendet werden können, die eine Klasse als Wert haben.

3.2 DAML+OIL

RDF und RDFS stellen eine sehr leichtgewichtige, einfach zu erlernende Möglichkeit dar, Semantik und Struktur in Webdokumente zu bringen. Die wichtigsten Merkmale sind die einfache Beschreibung von primitiven Aussagen und ihrer Bedeutung sowie die Möglichkeit, Informationen in Klassen und Subklassen einzuteilen und die Beziehungen zwischen diesen zu beschreiben. Mit XML, RDF und RDFS lassen sich einfache Ontologien erstellen, man könnte RDF also in etwa wie ein Assembler für Ontologien betrachten. Um jedoch Ontologien für wirkliche Anwendungen zu erstellen, fehlen Dinge wie komplexere Datentypen und ein weiteres Spektrum an Sprachkonstrukten, um eigene Klassen zu spezifizieren. Hier kommen zwei ursprünglich separate Ansätze ins Spiel, DAML [DAML Home, 2003] eine Erweiterung zu RDFS, die eine bessere Klassenspezifikation als RDFS ermöglicht. Und OIL [OIL Home, 2003] eine andere Initiative, die einen eher Logik-orientierten Ansatz wählte und ebenfalls eine Erweiterung der Möglichkeiten von RDFS zum Ziel hatte. Beide Ansätze gingen in DAML+OIL auf, dessen aktuellste Spezifikation [DAML+OIL, 2001] vom März 2001 stammt. Derzeit wird von der W3C WebOnt Arbeitsgruppe auf Basis dieser Sprache ein Standard namens *Web Ontologies Language (OWL)* erarbeitet.

Eine DAML+OIL Ontologie besteht im Prinzip aus RDF Tripeln. Diesen Tripeln werden jedoch spezielle Bedeutungen zugewiesen, so setzt sich das DAML+OIL Vokabular zusammen. Welche RDF Tripel das DAML+OIL Vokabular bilden, ist in der RDF Referenz [DAML+OIL, 2001] zu finden.

3.2.1 Datentypen

Im Gegensatz zu RDFS, wo Werte von Eigenschaften immer als Literale, also immer ohne Typisierung, angegeben werden, kann man in DAML+OIL auf Datentypen zurückgreifen die in XSDL spezifiert sind. Dies führt dazu, dass z.B: eine Produktnummer auch wirklich als Zahl interpretiert wird und nicht evtl. als String. Außerdem kann jeder Wert einer Eigenschaft auf bestimmte Grenzen limitiert werden und zwar nicht nur wie in RDFS einfach, sondern es können mehrere *ranges* definiert werden.

```
<daml:DatatypeProperty rdf:ID="age">
  <rdfs:comment>
    age is a DatatypeProperty whose range is xsd:decimal.
    age is also a UniqueProperty (can only have one age)
  </rdfs:comment>
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#nonNegativeInteger"/>
</daml:DatatypeProperty>
```

DAML+OIL beschränkt sich allerdings nicht auf einen Vorgegebenen Satz an Typen, sondern läßt Entwicklern freie Hand Eigene Typen zu definieren. Alle Literale, die nicht zur Property *daml:DatatypeProperty* gehören, müssen als *daml:ObjectProperty* in DAML+OIL oder RDFS definiert sein, und deren Werte werden auf bestimmte *ranges* limitiert.

3.2.2 Klassendefinitionen und Einschränkungen

DAML+OIL erweitert das Klassen und Vererbungsmodell von RDF bzw. RDFS, da die sprachlichen Mittel von RDFS nicht ausreichen um komplizierte Sachverhalte zu beschreiben. So kann man, wie bereits erwähnt, eigene Klassen definieren. Dabei erben Klassen Eigenschaften und Limits dieser Eigenschaften von ihren Superklassen bzw. vererben diese an ihre Subklassen. Außerdem bietet DAML+OIL ein breites Spektrum an Ausdrücken, um die Beziehungen von Klassen zu modellieren. Die wichtigsten davon sind:

unionOf die Vereinigung zweier Klassen, wobei eine neue Klasse entsteht.

disjointUnionOf Die disjunkte Vereinigung von zwei oder mehreren Klassen, z.B. die disjunkte Vereinigung aller Männer und Frauen bildet die Klasse der Personen.

intersectionOf Die Schnittmenge zweier Klassen z.B: Menschen $\sqcap$ Männer

complementOf Das Gegenteil z.B: weiblich $\neq$ männlich

isSameClassAs Zeigt Äquivalenz zu einer anderen Klasse auf was besonders wichtig ist, um Interoparabilität von verschiedenen Ontologien zu gewährleisten.

Alle Klassen wiederum können Eigenschaften haben. Diese können auf bestimmte Werte beschränkt werden. Zum Beispiel könnte es im Sortiment eines Sportartikelherstellers die Klasse der Rucksäcke geben und das Fassungsvermögen einer Rucksack Instanz kann von 1 Liter bis 80 Liter variieren.
DAML+OIL bietet natürlich noch viel mehr Sprachkonstrukte und Konzepte an, um Ontologien zu erstellen, auf die jedoch aus Platzgründen nicht weiter eingegangen werden kann.

Chapter 4

WebServices und Ontologien

Im Zusammenhang mit WebServices ist DAML+OIL[DAML+OIL, 2001] natürlich besonders interessant, da DAML+OIL Ontologien nicht nur verwendet werden können, um die Produkte eines Online Angebotes zu beschreiben, sondern auch um den WebService an sich zu beschreiben. Ontologien können die Automatisierung von WebServices vereinfachen oder sogar erst ermöglichen, denn durch ihre Computerlesbarkeit und Verständlichkeit kann der WebService und sein Interface in eindeutiger und sinnvoller Weise beschrieben werden.

4.1 DAML-S

DAML-S[DAML-S ,The DAML Service Coalition 2002] ist ein Zweig des DAML+OIL [DAML+OIL, 2001] Projektes und ist eine speziell für die Bedürfnisse von WebServices zugeschnittene Ontologie, geschrieben in DAML+OIL. DAML-S stellt Anbietern von WebServices einen Satz an Markup Konstrukten zur Verfügung, um die Eigenschaften, das Interface und die Produkte ihres Angebotes in eindeutiger Art und Weise zu beschreiben. Der Einsatz von DAML-S Markups in WebServices würde eine Reihe von Aufgaben wie das automatische Finden, das automatische Ausführen verbessern und die Interoparabilität von WebServices erhöhen. Diese Teilbereiche dienen auch als Struktur der Ontologie, die sich in drei Hauptkategorien einteilt.

Das Service Profil beschreibt einen Service nach außen und vereinfacht das Suchen und Finden von WebServices.

Das Prozeß Modell beschreibt detailliert die Funktionsweise eines WebServices.

Das 'Grounding' beschreibt das Kommunikationsinterface eines WebServices.

Alle drei Bereiche sind als DAML+OIL Klassen modelliert, und alle sind SubKlassen von *Service*, wobei Service einen echten WebService bezeichnen sollte. Allerdings gibt es keine Überprüfungsmechanismen, man könnte also eine einfache statische HTML Seite mit Service Markups versehen.

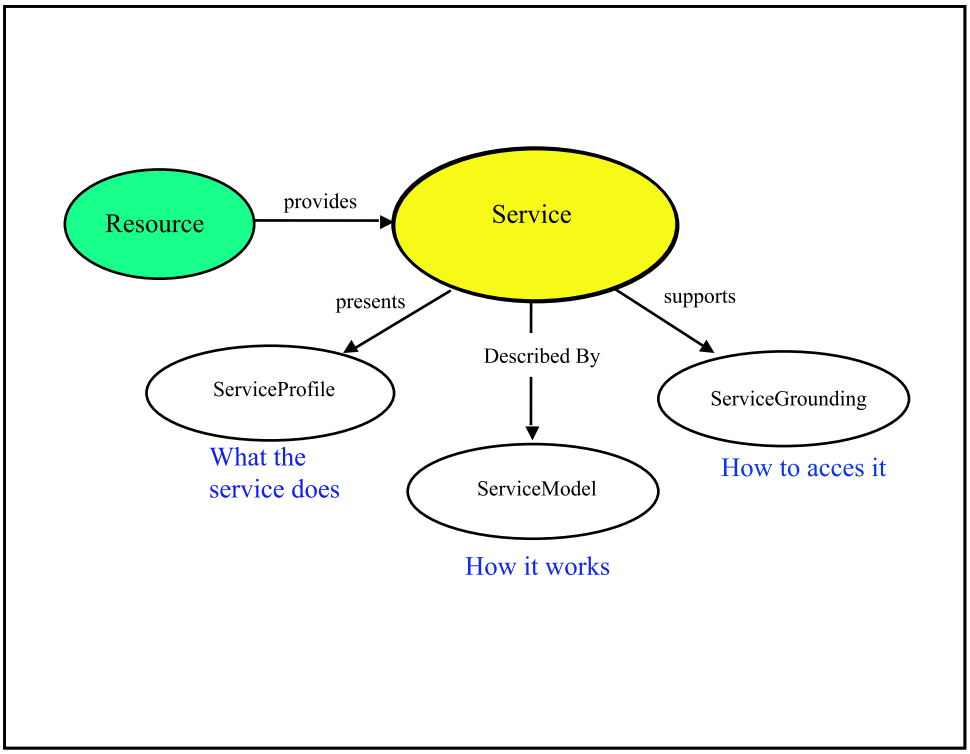


Figure 4.1: Die Top-Level Struktur eines DAML-S WebServices.

DAML-S definiert drei *Properties: presents, describedBy* und *supports*, die als *range* jeweils die drei obigen Klassen besitzen. D.h. ein *Service presents a ServiceProfile, is describedBy a ProcessModel, supports a ServiceGrounding*

4.1.1 Service Profil & Prozess Modell

Um zu erklären oder zu beschreiben, wie ein Service arbeitet, ist eine Notation nötig, um die Prozesse, Tätigkeiten und Modi der Durchführung zu beschreiben. DAML-S liefert eine Ontologie von Prozessen, modelliert durch eine Subklasse von *ServiceModel*. Die *ProcessModel* Kategorie besteht wiederum aus der *Process* Ontologie und der *ProcessControl* Ontologie. Wobei ersteres die Eingänge, die Ausgänge, die Vorbedingungen und die Seiteneffekte eines Services beschreibt. Letzteres erlaubt es, den (aktuellen) Zustand eines Prozesses zu beschreiben. Zusätzlich kann beschrieben werden, wie der Service aufgerufen werden kann und die Zustände bei seiner Beendigung.

Die *Process* Klasse ist das Basiselement der *Process* Ontologie. Ein *Process* kann eine beliebige Anzahl von Eingängen, Ausgängen, Vorbedingungen und Nachbedingungen sowie Seiteneffekte haben. DAML-S teilt *Process* in drei Arten: 1.*AtomicProcess*ist ein Prozeß,

der all seinen Input zur Aufrufzeit erhält, Berechnungen anstellt und alle Ergebnisse zur Endzeit zurück gibt.
2.*CompositeProcess* ist ein Prozeß, der aus mehreren atomaren Prozessen zusammengesetzt wird und auch wieder zerlegt werden kann.
3.*SimpleProcess* nennt man Prozesse, die nicht direkt aufgerufen werden können und auch nicht direkt mit einem speziellen *Grounding* verknüpft sind. *SimpleProcesses* bilden die Grundlage für Abstrakte Prozesse. Ein simpler Prozeß wird entweder von einem atomaren Prozeß realisiert oder zu einem zusammengesetzten Prozeß erweitert.

4.1.2 Service Grounding

Wenn der Service Abstrakt beschrieben wurde, muss das Ganze in echten, funktionierenden Code umgesetzt werden. Dafür ist die *ServiceGrounding* Klasse von DAML-S zuständig. Sowohl die Service- als auch die Prozeßmodelle sind abstrakte Beschreibungen des Services und seines Interfaces. Erst im *Grounding* werden diese abstrakten, ontologischen Aspekte in konkrete Spezifikationen umgesetzt. So wird hier spezifiziert, wie jeder atomare Prozeß (und somit auch zusammengesetzte Prozesse) mit Nachrichten angesprochen werden kann und somit auch benutzt werden können. Dabei lässt sich ein DAML-S *Grounding* direkt auf ein WebServices Description Language (WSDL) binding mappen.

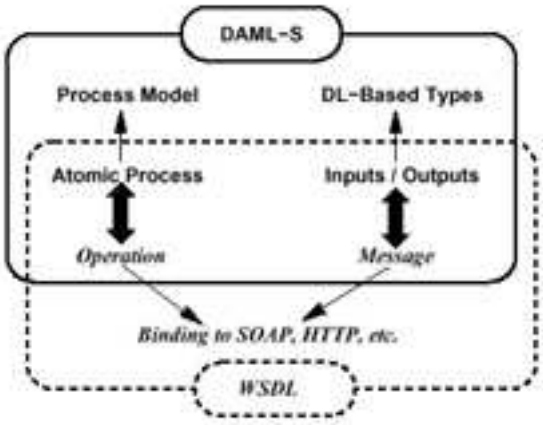


Figure 4.2: DAML-S und WSDL.

4.2 CongoBuy - ein fiktiver DAML-S WebService

Zum Abschluss wird hier noch ein Beispiel betrachtet. Es handelt sich dabei um den (fiktiven) WebService CongoBuy [DAML-S Coalition, 2002], der als Beispiel zum aktuellen DAML-S Release [Brickley, Guha 2003] auf den WebSeiten der DAML-S Koalition definiert

wurde. Bei CongoBuy handelt es sich um eine Sammlung von einzelnen WebServices (wie CongoLocateBook, CongoBillCreditCard usw.), die zu einem WebService zusammengefasst werden. CongoBuy verwendet viele DAML+OIL Ontologien, um sich selbst und Produkte zu beschreiben, insbesondere die *profile ontology* [1] und die *process ontology* [2]. Diese Ontologien bilden das Fundament um WebServices zu beschreiben. Von den dort definierten Klassen wird abgeleitet um einen speziellen Service wie CongoBuy zu beschreiben.

4.2.1 Die Atomaren Prozesse

Schritt eins bei der Definition eines Webservices ist die Deklaration der atomaren Prozesse, aus denen sich der gesamt Service zusammensetzt, was hier am Beispiel von *LocateBook* dargestellt wird. *LocateBook* erwartet einen Buchnamen als Input und liefert eine Beschreibung und den Preis zurück, falls das Buch sich im Katalog von CongoBuy befindet.

```
<daml:Class rdf:ID="LocateBook">
 <rdfs:subClassOf rdf:resource="&process;#AtomicProcess"/>
</daml:Class>

<rdf:Property rdf:ID="bookName">
 <rdfs:subPropertyOf rdf:resource="&process;#input"/>
 <rdfs:domain rdf:resource="#LocateBook"/>
 <rdfs:range rdf:resource="&xsd;#string"/>
</rdf:Property>
```

Der Input eines atomaren Prozesses kann obligatorisch oder optional sein. Etwas anders ist die Regelung der Ergebnisse, den Output, denn der Output hängt immer von bestimmten Konditionen ab. Der Output des LocateBook Services ist entweder die Beschreibung des Buches oder eine Fehlermeldung, wenn das Buch nicht im Sortiment ist. In DAML-S wird dieser Sachverhalt modelliert, indem die *range* des Outputs vom Typ *ConditionalOutput* ist, *ConditionalOutput* wiederum hat zwei *Properties* : 1.*coCondition* und 2.*coOutput*. Hängt das Ergebnis eines atomaren Prozesses nicht von Konditionen ab, wird die Kardinalität von *coCondition* auf 0 limitiert.

```
<daml:Restriction daml:cardinality="1">
 <daml:onProperty rdf:resource="#coCondition"/>
 <daml:hasValue rdf:resource="#condInCatalogueCondition"/>
</daml:Restriction>
```

Durch das Definieren der Ein- und Ausgänge der atomaren Prozesse sind die Voraussetzungen für ein automatisiertes Aufrufen und benutzen des Services geschaffen. Um nun auch eine automatische Komposition von Teil Services zu ermöglichen, müssen lediglich

[1] http://www.daml.org/services/daml-s/2001/10/Profile.daml
[2] http://www.daml.org/services/daml-s/2001/10/Process.daml

die Seiteneffekte des Prozesses definiert werden, sofern welche vorhanden sind. Mit diesen Voraussetzungen könnte aus den Prozessen *LocateBook, PutIntoCart* usw. die Spezifikation des WebServices CongoBuy automatisch generiert werden.

4.2.2 Die Groundings der Atomaren Prozesse

Um nun die abstrakten Definitionen ausführbar zu machen muss jeder Prozeß an genau ein *Grounding* gebunden werden. Das passiert über die Eigenschaft *hasGrounding*, die in der DAML-S Prozessontologie definiert ist.

```
<daml:Class rdf:about="LocateBook">
 <daml:sameClassAs>
  <daml:Restriction daml:cardinality="1">
   <daml:onProperty rdf:resource="#hasGrounding"/>
   <daml:hasValue rdf:resource="#LocateBookGrounding"/>
  </daml:Restriction>
 </daml:sameClassAs>
</daml:Class>
```

Das *Grounding* selbst ist nicht viel mehr als ein Mapping der in DAML-S definierten Ein- und Ausgänge auf WSDL Nachrichten, die dann vom zugrundeliegenden WSDL Binding implementiert werden, also z.B. ein HTML query, auf das eine Antwort zurückgegeben wird. Nachstehend ist ein Beispiel Grounding für den *LocateBook* Prozeß dargestellt, wobei die mit http://example.com gekennzeichneten Ressourcen auf ein hier nicht besprochenes WSDL Dokument verweisen.

```
<grounding:WsdlGrounding rdf:ID="LocateBookGrounding">
 <grounding:wsdlReference rdf:resource="http://www.w3.org/TR/2001/NOTE-wsdl-20010315">
  <grounding:otherReferences rdf:parseType="daml:collection">
   "http://www.w3.org/TR/2001/NOTE-wsdl-20010315"
   "http://schemas.xmlsoap.org/wsdl/soap/"
   "http://schemas.xmlsoap.org/soap/http/"
  </grounding:otherReferences>
  <grounding:wsdlDocuments rdf:parseType="daml:collection">
   "http://example.com/congo/congobuy.wsdl"
  </grounding:wsdlDocuments>
  <grounding:wsdlOperation rdf:resource="http://example.com//locatebook.wsdl#FindBook"/>
  <grounding:wsdlInputMessage rdf:resource="http://example.com/locatebook.wsdl#LocateBoo
   <grounding:wsdlInputMessageParts rdf:parseType="daml:collection">
    <grounding:wsdlMessageMap>
     <grounding:damlsParameter rdf:resource=#bookName>
      <grounding:wsdlMessagePart
```

```
       rdf:resource="http://example.com//locatebook.wsdl#BookName">
      </grounding:wsdlMessageMap>

... other message map elements ...

     </grounding:wsdlInputMessageParts>
     <grounding:wsdlOutputMessage
      rdf:resource="http://example.com/locatebook.wsdl#LocateBookOutput"/>
     <grounding:wsdlOutputMessageParts rdf:parseType="daml:collection">

... similar to wsdlInputMessageParts ...

     </grounding:wsdlOutputMessageParts>
<grounding:WsdlGrounding>
```

Chapter 5

Zusammenfassung und Ausblick

In dieser Ausarbeitung habe ich eine Einführung in das Themengebiet der Ontologien gegeben sowie deren Anwendungsmöglichkeiten im Zusammenhang mit dem SemanticWeb und WebServices betrachtet. Ausserdem habe ich Technologien, die für die Kombination WebServices/Ontologien wichtig sind, einführend behandelt, namentlich RDF, RDFS, DAML+OIL und DAML-S.
WebServices und nicht-kommerzielle Anwendungen des SemanticWeb haben eines gemeinsam: grosses, heterogenes und verteiltes Datenaufkommen, welches über alle Grenzen hinweg kompatibel gemacht und integriert werden muss. Das Internet und die zugrundeliegende Infrastruktur bieten die Basis, um diese Bedürfnisse zu befriedigen. Um zu veranschaulichen, dass das Internet in seiner heutigen Form diesen Bedürfnissen nicht gewachsen ist, kann man das Internet mit einer gewöhnliche Telefonleitung vergleichen. Eine Telefonverbindung alleine bringt keinen Nutzen, wenn die Gesprächspartner nicht die selbe Sprache sprechen. Ontologien können in dieser Misere als Übersetzungsservice dienen. Denn durch die Anreicherung von Daten mit Semantik kann auch eine Brücke zwischen zwei Interfaces, die nicht kompatibel sind, geschlagen werden. So können viele Teilnehmer untereinander kommunizieren, ohne sich auf eine einzige, unhandliche Datenbank, ja nicht einmal auf ein eindeutiges Vokabular einigen zu müssen.
Die zunehmende Standardisierung der nötigen Technologien und Sprachen durch das *W3C* und die Verbindung mit bereits bestehenden Standards (z.B: DAML-S und WSDL) wird die Akzeptanz und Verbreitung von Ontologien in der Web-Entwickler Communitiy erhöhen. Man könnte sich gut vorstellen, dass DAML-S in Verbindung mit einem PHP, Java oder anderem Highlevel-Sprachen Grounding Ontologien und WebServices praktisch sofort von einem rein akademischen Forschungsgebiet zu einer weitverbreiteten Technologie im Internet werden könnte.

Bibliography

[D.Fensel, 2000] Dieter Fensel: *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*; Springer-Verlag Berlin Heidelberg New York 2000

[Google] http://www.google.de; *http://www.google.de*

[Waller] Richard Waller: *How Big is the Internet*; http::://www.waller.co.uk/web.htm

[Franz Embacher] Franz Embacher: *Bewertung von Webseiten durch Google*; http://www.ap.univie.ac.at/users/fe/Lehre/aussermathAnw/Google.html

[TimBray et al., 2000] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, and Eve Maler, editors: *Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation 6 October 2000*; http://www.w3.org/TR/2000/REC-xml-20001006

[Lassila, 1998] Ora Lassila: *Web Metadata: A Matter of Semantics*; IEEE Internet Computing 2(4): 30-37 (1998)

[Principia Cybernetica Web] Principa Cybernetica Web: *Principia Cybernetica, Ontology Introduction*; http://pespmc1.vub.ac.be/ONTOLI.html

[Gruber] Tom Gruber: *What is an Ontology*; http://www-ksl.stanford.edu/kst/what-is-an-ontology.html, Stanford University

[Dutra, 2001] Mark Dutra: *Ontologies for WebServices*; http://www.omg.org/news/meetings/workshops/webservices_2002.htm, 2002

[Weibel et al., 1995] S.Weibel,J.Gridby and E.Miller: *OCLC/NCSA MetaData Workshop Report*; Dublin, 1995. http://www.oclc.org:5046/oclc/research/conferences/metadata/dublin_core_rep

[Berners-Lee et al., 2001] Tim Berners-Lee, James Hendler, Ora Lassila: *The Semantic Web*, Scientific American, May 2001

[Lassila & Swick, 1999] Ora Lassila, Ralph R. Swick: *Resource Description Framework (RDF) Model and Syntax Specification W3C Recommendation 22 February 1999*; W3C 1999. http://www.w3.org/TR/REC-rdf-syntax/

[Brickley, Guha 2003] Dan Brickley, R.V. Guha: *RDF Vocabulary Description Language 1.0: RDF Schema*; W3C 2003. http://www.w3.org/TR/rdf-schema/

[DAML Home, 2003] The Defense Advanced Research Projects Agency: *The DARPA Agent Markup Language (DAML)*; http://www.daml.org/

[DAML+OIL, 2001] Ian Horrocks, Frank van Harmelen, Peter Patel-Schneider et al: *DAML+OIL (March 2001)*; http://www.daml.org/2001/03/daml+oil-index

[OIL Home, 2003] The OntoKnowledge: *The Ontology Inference Layer (OIL)*; http://www.ontoknowledge.org/oil/

[DAML-S ,The DAML Service Coalition 2002] The DAML Services Coalition (alphabetically Anupriya Ankolenkar, Mark Burstein, Jerry R. Hobbs, Ora Lassila, David L. Martin, Drew McDermott, Sheila A. McIlraith, Srini Narayanan, Massimo Paolucci, Terry R. Payne and Katia Sycara): *DAML-S: Web Service Description for the Semantic Web*; The First International Semantic Web Conference (ISWC), Sardinia (Italy), June, 2002.

[DAML-S Coalition, 2002] The DAML-S Coalition: *DAML-S 0.7 Draft Release*; http://www.daml.org/services/daml-s/0.7/, 2002.

[Fellbaum, 1999] Christiane Fellbaum (ed.): *WordNet: An Electronic Lexical Database*, MIT Press, 1999.

[Clark, 2003] Kendall Grant Clark: *The true meaning of service*, O'Reilly webservices.xml.com, http://www.xml.com/pub/a/ws/2002/07/17/daml-s.html?page=1 , 2003.