Matchmaking

Seminararbeit

vorgelegt am
Lehrstuhl für Praktische Informatik IV
Prof. Dr. W. Effelsberg
Fakultät für Mathematik und Informatik
Universität Mannheim

August 2004

Von
Joachim Jacobi
aus
Heidelberg

Betreuer: Marcel Busse

Inhaltsverzeichnis

Seite

Abkürzungsverzeichnis

DAML:	DARPA Agent Markup Language
DAML-S:	DARPA Agent Markup Language for Services
RDF:	Ressource Description Framework
RDF-S:	RDF Schema
OIL:	Ontology Interface Language
XML:	Extensible Markup Language
WWW:	World Wide Web

Abbildungsverzeichnis

1 Einleitung

Diese Ausarbeitung ist Teil des Teleseminars über „Web Services“, das in Kooperation der Lehrstühle von Prof. Dr. W. Effelsberg, Universität Mannheim sowie von Prof. Dr. H. Schmeck und Prof. Dr. H. Hartenstein, beide Universität Karlsruhe, abgehalten wurde und handelt von dem Thema „Matchmaking in Web Services“. Das Ziel dieses Teleseminars, neben der Diskussion von Grundlagen und Anwendungen von Web Services, war, dass die Studenten die Verwendung von Werkzeugen wie BSCW (Basic Support for Cooperative Work) erlernen. Die Themen wurden soweit möglich jeweils auf einen Karlsruher und einen Mannheimer Studenten aufgeteilt. Beide hatten einen gemeinsamen Vortag gehalten, jedoch sollte die Seminararbeit getrennt angefertigt werden und das im Seminar jeweils vorgetragene (Teil-) Thema vertiefen.

Web Services sind alle Anwendungen oder Anwendungskomponenten, die über Standard-Web-Protokolle aufgerufen werden können. Matchmaking in Web Services befasst sich mit dem Suchprozess, der durchgeführt werden muss, damit ein nach einem Web Service Suchender die gewünschte Anwendung findet.

Mein Mannheimer Kommilitone Stefan Graber hat im ersten Teil der Seminararbeit die bereits standardisierten Möglichkeiten des Matchmaking, nämlich die momentan verwendeten keywordbasierten und framebasierten Suchsysteme, kritisch betrachtet und Nachteile dieser Methoden herausgearbeitet.

Es zeigte sich dabei vor allem der Wunsch, sowohl schneller und präziser die gewünschten Informationen oder Ressourcen finden zu können, aber auch den Inhalt von Ressourcen für Maschinen verständlich zu machen. Damit könnte in Zukunft die Suche nach und der Gebrauch von verfügbaren Informationen und Ressourcen auf Maschinen übertragen werden.

Dieser zweite Teil der Seminararbeit befasst sich mit dem Ansatz des „Semantic Web“, der mit Hilfe von Ontologien den Inhalt von Daten, insbesondere die Funktion von Web Services genauer definiert und erläutert, warum dieser Weg eine befriedigende Lösung bieten kann. Vertiefend wird hierbei auf DAML-S und DAML+OIL als „Ontology Languages'“ eingegangen. Anhand eines Beispiels wird die Struktur einer Ontologie erklärt; anhand von DL bzw. RACER wird daraufhin erläutert, wie es zu einem „Match“ kommt. Eine kritische Würdigung dieser Ansätze rundet die Arbeit ab.

2 Semantic Web

Das Semantic Web ist eine Weiterentwicklung des heutigen WWW. Während im WWW der Fokus auf Präsentation und Syntax gerichtet ist, werden im Semantic Web Informationen mit wohldefinierten Bedeutungen versehen. Dadurch soll die Kooperation zwischen Mensch und Maschine aber auch die Kommunikation zwischen Maschine und Maschine verbessert werden. Jedoch darf man dabei nicht an eine weiterentwickelte Stufe der künstlichen Intelligenz denken; die Maschinen operieren mit ihrem jetzigen Entwicklungsstand weiterhin auf Daten, welche vom Menschen annotiert[1] wurden, auch wenn sie in bedingtem Maße dazu fähig sind, durch Inferenz[2] neues Wissen zu erschließen. Damit das Semantic Web erfolgreich ist, müssen die Maschinen Zugang zu strukturierten Archiven haben, wo sie eindeutige Informationen und Spezifikationen finden [8].

3 RDF und RDFS

3.1 RDF

Einer der ersten Ansätze, den Inhalt von Daten auch für Maschinen verständlich zu machen, war das „Ressource Description Framework“ RDF, ein Standard (Recommendation) des W3C, der im Februar 1999 verabschiedet wurde. RDF dient als formales Modell zur Definition, Beschreibung und zum Austausch von Metainformationen von Ressourcen. Somit erweitert es XML, welches zur Beschreibung von Struktur und Syntax benutzt wird, um die Komponente der Semantik für Ressourcen [2].

Das Prinzip von RDF ist einfach und übersichtlich gehalten, das Datenmodell besteht aus drei Objekttypen:

Ressourcen (Resources): Alle Entitäten in RDF sind Ressourcen, vorausgesetzt, sie können mit einer URI beschrieben werden. So kann eine gesamte Webseite oder nur ein Element aus einem HTML- oder XML- Dokument, aber auch ein Buch oder eine Zeitschrift eine Ressource darstellen.

---

[1] Annotation: Anreicherung von Dokumenten mit Metadaten basierend auf einer Ontologie

[2] Durch Schlussfolgerung abgeleitetes neues Wissen

Eigenschaften (Properties): Ressourcen werden durch spezielle Eigenschaften definiert oder beschrieben. Eine Eigenschaft ist ein spezieller Aspekt, ein Attribut oder eine Beziehung, die eine Ressource beschreibt oder besitzt.

Aussagen (Statements): Eine Ressource bildet zusammen mit einer Eigenschaft und einem Wert für diese Eigenschaft eine Aussage. Aus Subjekt (Ressourcen), Prädikate (Eigenschaften) und Objekte (Werte dieser Eigenschaften) werden Tripel gebildet. Ein Objekt kann eine Ressource (URI) oder ein so genanntes Literal (String) sein [5].

Abbildung 1 veranschaulicht ein solches Tripel. Es stellt eine Aussage dar, die wiederum als eine Ressource verwendet werden kann.

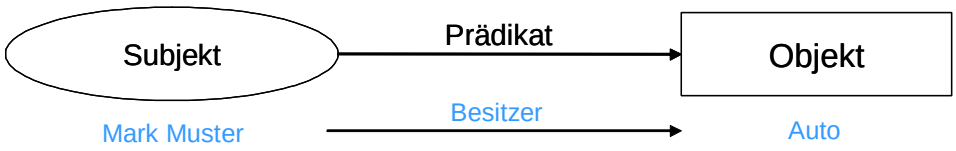

Abbildung 1. Ein RDF Tripel (3-Tupel)

Am Beispiel der Bedeutung von Metadaten für Mensch und Maschine soll der Sinn von RDF aufgezeigt werden. Die beiden Sätze "Mark Muster ist Eigentümer des roten Autos" und "Der Eigentümer des roten Autos ist Mark Muster" unterscheiden sich in Ihrer Bedeutung für den Menschen nicht. Eine Maschine kann jedoch nicht erkennen, dass der Inhalt des ersten Satzes mit dem des zweiten übereinstimmt und sich die Sätze nur dadurch unterscheiden, dass der erste aktiv, der zweite passiv formuliert ist. Mit Hilfe von Metadaten können die einzelnen Teile nun in Beziehung zueinander gebracht werden und ihre Zugehörigkeit definiert werden. Zur Darstellung von Aussagen die in RDF verfasst wurden werden Pfeildiagramme (s. Abbildung 1) verwendet [2].
Das Objekt stellt den Wert des Subjekts dar; die Beziehung der beiden Objekte zueinander wird durch einen Pfeil, der die Eigenschaft dieser Beziehung beschreibt, dargestellt. Der Wert des Subjekts kann je nach Ausprägung ein Literal oder eine Ressource sein.

3.2 RDFS

Wie im Abschnitt 3.1 dargestellt, werden mit Hilfe von RDF Ressourcen mit bestimmten Eigenschaften verbunden und Beziehungen zwischen verschiedenen Ressourcen beschrieben. RDFS (RDF Schema) stellt eine Erweiterung zu RDF dar, so dass auch Eigenschaften oder Beziehungen zwischen den Eigenschaften, mit deren Hilfe Ressourcen beschrieben werden sollen, definiert werden können.

RDF Schema stellt ein Standardvokabular zur Typisierung von RDF Ressourcen dar. Es bietet definierte Konzepte zur Beschreibung von Klassen von Ressourcen und Eigenschaften, sowie deren Zusammenhänge.

So kann nun zusätzlich zum Beispiel aus Abbildung 1 eine weitere Definition hinzugefügt werden, die besagt, dass der Wert der Eigenschaft „Besitzer“ vom Typ „Person“ sein muss. Diese Bildung von Klassen und Subklassen ist vor allem nützlich bei der Suche nach Daten, da die vorhandenen Metadaten weitere Rückschlüsse zulassen. Im Beispiel besteht nun die Möglichkeit, dass nur noch Personen gefunden werden, die in die Klasse „Besitzer“ eingeteilt sind.

Mit RDFS können auch von einem anderen RDF Schema Eigenschaften geerbt werden, und es wurde eine einfache Ausprägung für die Definition von Wertebereichen und Definitionsbereichen eingeführt. Allerdings kann eine Eigenschaft höchstens eine Wertebereichs- und eine Definitionsbereichseigenschaft besitzen [2].

4 Ontologien

Ontologie ist ursprünglich ein Begriff aus der Philosophie. Er umschreibt die Lehre vom Sein, beschäftigt sich mit der Natur und Organisation der Realität [7]. In der Informatik versteht man unter Ontologie eine explizite formale Spezifikation einer gemeinsamen

Konzeptualisierung (orig.: „shared conceptualization“) [3]. Sie beschreibt einen Wissensbereich (knowledge domain) mit Hilfe einer standardisierenden Terminologie sowie durch Beziehungen und Ableitungsregeln zwischen den dort definierten Begriffen.

Eine Ontologie beschreibt ein allgemeines Vokabular in einer Domäne und die Bedeutung einzelner Einheiten in diesem Vokabular, um sowohl für Menschen als auch für Maschinen den Inhalt von Ressourcen verständlich zu umschreiben.
Im Zusammenhang mit dem Semantic Web wird eine Ontologie verstanden als ein formales, semantisches Modell, welches verwendet wird, um den Austausch von Wissen zwischen Mensch und Maschine zu ermöglichen, indem Daten interpretiert und auch zueinander in Beziehung gesetzt werden können. Im semantischen Web wird nicht die Erschaffung einer einzigen Ontologie angestrebt, vielmehr kann je nach Kontext eine eigene Ontologie definiert werden. Es ist auch durchaus denkbar, dass eine Ontologie verschiedene Definitionen hat [2].

5 DAML+OIL und DAML-S

5.1 DAML+OIL

Der oben beschriebene Standard RDFS bietet zwar sehr eingeschränkt die Möglichkeit, Ontologien zu beschreiben, dennoch fehlen wesentliche Bestandteile, um der Mächtigkeit von Ontologien gerecht zu werden. So ist zum Beispiel das Verwenden von logischen Operatoren in RDF nicht möglich.
DAML: Die „DARPA Agent Markup Language“ wurde im Rahmen eines Forschungsprogramms der dem US-Verteidigungsministerium unterstellten „Defense Advanced Research Projects Agency“ entwickelt. Das DAML-Programm begann im August 2000, seine Ergebnisse sind aufgrund des „unclassified“-Status des Projekts allgemein zugänglich, werden aber von der DARPA durchaus auf militärische Anwendungszwecke hin verwendet [5].
OIL: Die Anfänge der von der EU unterstützten Entwicklung der Sprache *Ontology Interface Language* (OIL) liegen im August 1999, allerdings mit abweichender Zielsetzung, die eher dem E-Commerce-Hype der 1990-er entsprang. Die Sprache OIL war zwar für die Formulierung von Ontologien geeignet, jedoch nicht XML-konform und für die Integration in (X)HTML-Seiten ungeeignet [5].

DAML+OIL: Sowohl DAML als auch OIL waren von Beginn an auf einem hohen Abstraktionsniveau angesiedelt, jedoch war OIL im Gegensatz zu DAML auf die Formulierung von Ontologien ausgerichtet, was bei DAML erst in der speziellen Version DAML-ONT der Fall war. Offensichtlich erkannten die Beteiligten jedoch bald die Gemeinsamkeiten der beiden Projekte und spezifizierten sie als Zusammenfassung beider Sprachen DAML+OIL. Diese ist ein wichtiger Bestandteil der Semantic-Web-Initiative zur Definition von Ontologien, welche ein Kernstück des Semantic Webs darstellen [5].
DAML+OIL ermöglicht die Definition von Klassen, Subklassen, Eigenschaften und Subeigenschaften. Sie baut auf den Beschreibungssprachen XML und RDF auf und erweitert diese um die Möglichkeit, komplexe Beziehungen darzustellen und Vererbungen von Eigenschaften zu realisieren. Mit DAML+OIL ist es möglich, unterschiedliche Ontologien zu entwickeln und diese miteinander zu verknüpfen. Der Standard besitzt eine klar definierte Semantik und ist somit eine von Maschinen interpretierbare Sprache.

Ein Auszug der wichtigsten Neuerungen im Vergleich zu RDFS [2]:

- Darstellung von Definitions- und Wertebereichen
- Formulierung von Kardinalitäten (Beispiel: x ist genau dann eine Instanz der definierten Klasse, wenn es N eindeutige Werte y gibt, so dass (x, y) eine Instanz von P ist (P _Eigenschaften))
- Relationen auch über Klassen und Hierarchiestufen hinweg
- Umfassendere Beschreibung von Eigenschaften, diese können zum Beispiel als „lokal“ definiert werden
- Äquivalenzausdruck um zu zeigen, dass zwei Instanzen einer Klasse oder einer Eigenschaft gleich sind
- Mit logischen Operatoren können Axiome gebildet werden (Teilmenge, Vereinigungsmenge, Komplement)

Das W3C-Konsortium ist bemüht um die Verabschiedung eines Sprachstandards für die

Beschreibung von Ontologien. Dieser Sprachstandard wird den Namen OWL (Web Ontology Language) tragen und liegt zum Zeitpunkt des Verfassens dieser Arbeit als „Candidate Recommendation“ dem W3C vor. Die Unterschiede zu DAML+OIL sind minimal und deshalb wird auch eine große Umstellung der bestehenden DAML+OIL Dokumente nach OWL ausbleiben.

5.2 DAML-S

Die Beschreibungssprache DAML-S (DARPA Agent Markup Language for Services) ist eine DAML+OIL Ontologie, welche mit spezifischen Erweiterungen für die semantische Beschreibung von Web Services eingesetzt werden kann. DAML-S ist in drei Teile gegliedert (Abbildung 2):

1. Zum einen wird das **Service Profile** für das Anpreisen und das Auffinden von Services verwendet und wird oft als Eintrag in den Gelben Seiten dargestellt. Der Inhalt des Service Profiles umfasst die Beschreibung der eigentlichen Funktion des Services, dessen Ein- und Ausgabe und seine Voraussetzungen und Auswirkungen.
2. Als zweiter Teil gibt das **Service Model** eine genaue Beschreibung des Ablaufs des Services wieder, umfasst also die Definition des Kontroll- und Datenflusses.
3. Der dritte Teil, das **Service Grounding**, umschreibt detailliert die Art und Weise, wie der beschriebene Service mit anderen Services über Nachrichtenaustausch kommuniziert. Die verwendeten Nachrichten werden in WSDL definiert [2].

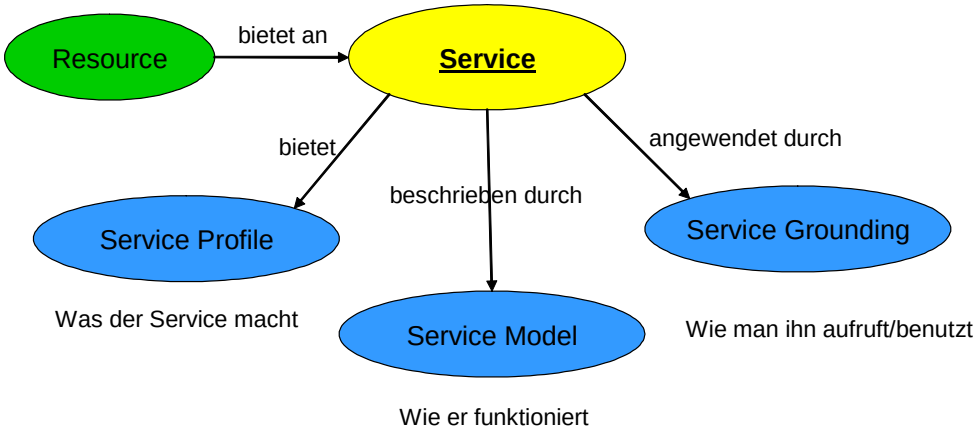

Abbildung 2. Die Service Ontologie von DAML-S

6 Beispiel-Ontologie zur Beschreibung eines Web Services

Zur Illustration einer Ontologie zur Beschreibung eines Services habe ich folgendes Szenario entworfen: Die Firma „Bleicker Schrauben + Dübel GmbH“ möchte den Vertrieb ihrer produzierten Schrauben über das Internet standardisieren. Damit soll ermöglicht werden, dass Kunden (Nach-)Bestellungen von Schrauben automatisieren können und ihr Computer selbstständig die Firma finden, kontaktieren und Bestellungen aufgeben kann.

Zur formalen Beschreibung des Beispiels werde ich im Folgenden *Description Logic (DL)* verwenden, ein Formalismus, der äquivalent zu DAML-OIL die Definition von Klassen, Subklassen, Eigenschaften und Subeigenschaften ermöglicht. Mit Hilfe der Description Logic lässt sich die zu beschreibende Ontologie jedoch übersichtlicher und verständlicher darstellen als mit der DAML-OIL Syntax.

Ich benutze in meiner Ontologie die von DAML-S vordefinierte Klasse *Profile* als gemeinsame Superklasse der Klassen *ServiceProfile, Advertisement* und *Request*, diese können wie folgt dargestellt werden.

$$\begin{aligned} \text{Profile} &\subseteq \top \\ \text{ServiceProfile} &\subseteq \text{Profile} \\ \text{Advertisement} &\subseteq \text{Profile} \\ \text{Request} &\subseteq \text{Profile} \end{aligned}$$

Weiterhin habe ich zwei Arten von Services in dieser Ontologie definiert: *Sales* und *Delivery*. *Sales* beschreibt den Verkauf eines Elements der Klasse *Equipment* durch Beschränkung der Objekt- und Datentyp-Eigenschaften wie zum Beispiel die Einheit Preis. In Übereinstimmung mit der Version DAML-S 0.6 nehme ich ebenso die anbietenden und nachfragenden Akteure als Wert der *providedBy* und *requestedBy* Eigenschaften mit auf.

$$
\begin{aligned}
\text{Sales} \subseteq\ & (=_1 \text{providedBy.Actor}) \cap \\
& (=_1 \text{requestedBy.Actor}) \cap \\
& (=_1 \text{item.Eqipment}) \cap \\
& (=_1 \text{hasQuantity.positiveInteger}) \cap \\
& (=_1 \text{canDeliver.Delivery}) \cap
\end{aligned}
$$

$$
\begin{aligned}
\text{Delivery} \subseteq\ & (=_1 \text{location.Delivery.Location}) \cap \\
& (=_1 \text{date.DeliveryDate}) \cap
\end{aligned}
$$

$$
\begin{aligned}
\text{Actor} \subseteq\ & (=_1 \text{hasName.ActorName}) \cap \\
& (=_1 \text{hasCreditLevel.Integer}) \cap
\end{aligned}
$$

Delivery beschreibt die Struktur der Lieferbedingungen durch Spezifizierung. Es darf zum Beispiel nur genau ein Lieferdatum und genau eine Lieferadresse geben.

Um den Bergriff Schraube zu definieren habe ich eine Klasse *Screw* als Subklasse der Klasse Equipment eingeführt, die einige Eigenschaften wie Schrauben - Typus, Länge und Packungs-größe besitzt.

$$
\begin{aligned}
\text{Screw} \subseteq\ & \text{Equipment} \\
& (=_1 \text{hasSort.Sort}) \cap \\
& (=_1 \text{hasSize.Size}) \cap \\
& (=_1 \text{hasPackagingSize.PackagingSize}) \\
\text{Sort} \equiv\ & \text{Holzschrauben} \cup \text{Blechschrauben} \\
\text{Size} \equiv\ & 1 \cup 2 \cup 2 \cup ... \cup 10
\end{aligned}
$$

Wie in Abschnitt 5.2 beschrieben wird ein Service durch die Ein- und Ausgabeeigenschaften des Service-Profils dargestellt

Die Eigenschaften der Eingabe beschreiben die Information, die ein Service benötigt, um die nötigen Berechnungen auszuführen. In diesem Beispiel wäre das die Packungsgröße oder der Preis. Die Ausgabe gibt das Ergebnis des Vorgangs des Services an. Beim Verkauf

von Schrauben könnte das die Bestätigung des Lieferauftrags sein, die die zu liefernde Menge und den Ort, an den geliefert wird, beschreibt.

In dieser Arbeit werden diese Bestandteile in Inputs und Outputs unterschieden, bezogen auf den Zusammenhang, in dem sie verwendet werden. Inputs werden von Kunden und der Firma dazu benutzt, die Eigenschaften des jeweiligen Handels zur beschreiben, also Menge, Preis und Lieferadresse. Outputs sollen das Produkt selbst beschreiben.

$$\begin{aligned} \text{inputs} &\subseteq \text{parameter} \\ \text{outputs} &\subseteq \text{parameter} \end{aligned}$$

$$\begin{aligned} \text{hasQuantity} &\subseteq \text{inputs} \\ \text{hasUnitPrice} &\subseteq \text{inputs} \\ \text{canDeliver} &\subseteq \text{inputs} \end{aligned}$$

$$\text{item} \subseteq \text{outputs}$$

6.1 Advertisement

Ein Beispiel eines Angebots (Advertisements) wäre folgendes: Angenommen die Firma „Bleicker Schrauben + Dübel GmbH“ möchte Schrauben verkaufen. Die Eigenschaften in Bezug auf *Sales* und *Delivery* könnten wie folgt aussehen:

- Name: Bleicker S+D GmbH
- Schrauben
- Mindestens 5000 Stück pro Lieferung
- 20 € /1000 Stück
- Käufer hat eine Kredit-Würdigkeit von mindestens 5
- Lieferung innerhalb von zwei Tagen

In der Schreibweise der *Description Logic* könnte das Angebot so aussehen:

$$
\begin{aligned}
\text{Advert1} \equiv\ & (\text{providedBy}(\text{Actor} \cap \forall\, \text{hasName.Bleicker}),\\
& \text{requestedBy}(\text{Actor} \cap \geq_{5}\, \text{hasCreditLevel}),\\
& \text{profile}(\text{ServiceProfile} \cap\\
& \text{Sales} \cap\\
& \forall\, \text{item.}(\text{Screw} \cap \geq_{1000}\, \text{PackagingSize}) \cap\\
& \leq_{20}\, \text{hasUnitPrice} \cap\\
& \geq_{5000}\, \text{hasQuantity}\\
& \forall\, \text{delivery.}(\text{Delivery} \cap\\
& \forall\, \text{deliveryDate.}(\leq_{2}\, \text{afterOrder})))
\end{aligned}
$$

6.2 Request

Entsprechend wird eine Anfrage (Request) definiert, in der ein Kunde Schrauben kaufen möchte. Die Angaben bezüglich *Sales* und *Delivery* könnten so aussehen:

- 10000 Blechschrauben der Größe 5 cm
- der Preis: weniger als 300 €, also weniger als 30 €/Packung

Aus Sicht der *Description Logic* sind Request und Advertisement fast identisch, beide sind der Klasse *ServiceProfile* untergeordnet.

$$
\begin{aligned}
\text{Request1} \equiv\ & \text{profile}(\text{ServiceProfile} \cap\\
& \text{Sales} \cap\\
& \forall\, \text{item.}(\text{Screw} \cap \forall\, \text{hasSort.Blechschraube} \cap\\
& \text{hasSize.5} \cap\\
& \forall\, \text{item.}(\text{Screw} \cap \geq_{1000}\, \text{PackagingSize}) \cap\\
& \leq_{30}\, \text{hasUnitPrice})))
\end{aligned}
$$

7 Definition Matchmaking

Matchmaking wird hier als ein Prozess definiert, der einen Host benötigt, dort eine Anfrage oder ein Angebot als Eingabe verarbeitet und als Rückgabe alle Angebote ausgibt, die den Anforderungen entsprechen, die in der Eingabe beschrieben werden [6].
Formal: Sei α die Menge aller gespeicherten Angebote (Services). Für eine Eingabe Q (Query/Anfrage) gibt der Matchmaking Algorithmus des Hosts die Menge aller kompatiblen Angebote zurück, *matches(Q)* [6]:

$$Matches\ (Q) = \{A \in \alpha \mid compatible\ (A,Q)\ \}$$

Zwei Beschreibungen (Strukturen) sind kompatibel, wenn die Schnittmenge beider Strukturen nicht leer ist [6]:

$$compatible\ (D1, D2) \Leftrightarrow \neg\ (D1 \cap D2 \subseteq \bot)$$

Im oben erläuterten Beispiel wäre die Schnittmenge der Anfrage (*Request1*) geschnitten mit dem Angebot (*Advert1*) ausreichend. Formal [6]:

$$Advert1 \in matches\ (Request1)$$

Dabei ist zu beachten, dass, auch wenn in der Anfrage keine Angaben über die Lieferung gemacht werden, beide Beschreibungen kompatibel sind. Es muss folglich nicht zu jeder Eigenschaft eine Angabe gemacht werden. Es reicht aus, wenn Teile der Informationen in beiden Beschreibungen kompatibel sind.
Um den Begriff Matchmaking noch besser zu erläutern, möchte ich im Folgenden auf eine bereits existierende Inferenz – Maschine eingehen, dem „*RACER DL Reasoner"*.

8 RACER DL Reasoner

RACER ist eine Semantic Web Inferenz – Maschine zur Entwicklung von Ontologien und zur Bearbeitung von Anfragen über RDF Dokumente sowie in Bezug auf RDFS/DAML- und OWL-Ontologien.

Weiterhin ist RACER ein System zur Analyse und Schlussfolgerung für die *Description Logic (DL)*. Es verwendet dabei sogenannte *ABoxes* und *TBoxes*, eine Gliederung, die in der Architektur eines Standard DL Systems verwendet wird [4].

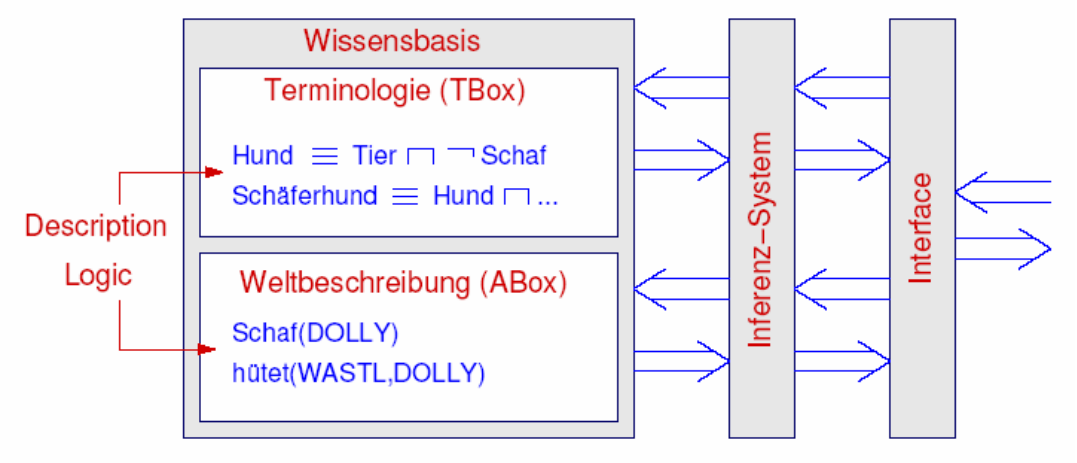

Abbildung 3: Architektur eines Standard DL Systems [1]

8.1 TBox (Terminologien)

Eine Komponente der Wissensbasis eines DL Systems ist die TBox. Sie besteht aus einer Menge von terminologischen Axiomen, welche Aussagen darüber machen, wie Konzepte oder Rollen zueinander in Beziehung stehen. Spezielle Axiome sind Definitionen, mit denen man Symbolnamen für komplexe Beschreibungen einführen kann. Die Symbole einer Terminologie sind unterteilt in Namenssymbole N_T und Basissymbole B_T. Namenssymbole sind Symbole, die atomare Konzepte auf der linken Seite einer Definition sind, Basissymbole sind Symbole, die nur auf der rechten Seite einer Definition auftreten [1] .

8.2 ABox (Weltbeschreibungen)

Die ABox ist eine weitere Komponente eines wissensbasierten Systems, in dem eine *Description Logic* - Sprache zur Anwendung kommt. Sie besteht aus einer endlichen Menge von Aussagen über Individuen. Eine Aussage legt die Zugehörigkeit eines Individuums zu Konzepten fest, bzw. bestimmt die Rollen-Beziehung zweier Individuen, z.B. Schaf (DOLLY), hütet (WASTL, DOLLY).
Somit erlaubt die ABox die Formulierung einer konkreten Situation bezüglich des terminologischen Rahmens der TBox und wird deshalb auch als Weltbeschreibung bezeichnet.
Durch die getrennte Argumentation über die Struktur und über Individuen erreicht man eine Verminderung der Komplexität von Interferenzen, was eine Verkürzung der Laufzeit mit sich bringt [1].

8.3 Matching Algorithmus

Um den von RACER verwendeten Matching Algorithmus zu verstehen, muss zuerst die Definition „Level eines Matches“ eingeführt werden. Dieser Begriff ist nützlich, da es nicht unbedingt ausreicht zu ermitteln, dass ein Advertisement (Angebot) und ein Request (Anfrage) nicht inkompatibel sind. Deshalb wird der Wert „Schnittmenge ausreichend“ auf verschiedene Level von Matches erweitert:

Exact :	Request R und Adv. A haben das gleiche Konzept	formal:	$A \equiv R$
PlugIn:	Request R ist ein Subkonzept Adv. A	formal	$R \subseteq A$
Subsume:	R ist ein Superkonzept von A	formal	$A \subseteq R$
Intersection:	Die Schnittmenge von A und R ist ausreichend	formal:	$\neg(A \cap R \subseteq \bot)$
Disjoint:	sonst (keine Schnittmenge vorhanden)	formal:	$A \cap R \subseteq \bot$

Am Anfang des Matchmaking Prozesses wird RACER mit einer Service Ontologie, wie sie zum Beispiel in Abschnitt 6 beschrieben wurde, initialisiert. Diese wird RACER dann zur Berechnung von Schnittmengen zwischen Angebot und Nachfrage während des gesamten Matchmaking-Prozesses benutzen. Von jedem gespeicherten Angebot wird das Service Profile an RACER gesendet. Dieser erstellt eine Service-Profile-Hierarchie, in die er alle Angebote aufnimmt.

Empfängt Racer eine Anfrage, stuft er ihr Service Profile ein und errechnet die Schnittmengen zwischen der Anfrage und allen vorhandenen Angeboten. Dann ordnet er die errechneten Schnittmengen nach dem Level der Matches ein und gibt sie der Reihenfolge nach, dass heißt zuerst die Matches des Levels Exact, danach die des Levels PlugIn usw., zurück [6].

9 Zusammenfassung und Ausblick

Mit Hilfe von RDF ist es möglich, eine Struktur der Daten zu erstellen. Einen entscheidenden Schritt weiter geht der Ansatz der Prozess-Ontologien, der es erlaubt, Daten mit Struktur und Bedeutung zu versehen und Beziehungen mit anderen Daten herstellen zu können. Der Vorschlag, DAML+OIL als Standard einzuführen, wurde vom W3C dahingehend abgeändert, dass eine „Web Ontology Language (OWL)“ entwickelt wurde, die sich jedoch kaum von DAML+OIL unterscheidet. Die Entscheidung, ob OWL als Standard verwendet wird, ist noch nicht gefallen.

Es ist eine durchaus vorstellbare Möglichkeit, dass sich durch den Einsatz von Ontologien der Gebrauch des Internets revolutionieren kann und die Suche nach Ressourcen nicht nur erleichtert, sondern auch den Einsatz von Software zur Automatisierung von Such- und Anwendungsprozessen für das Internet ermöglicht.

Man muss jedoch auch berücksichtigen, dass es momentan noch sehr schwer ist, Ontologien zu erstellen. Damit sich der Ansatz des Semantic Webs unter Verwendung von Prozess-Ontologien durchsetzen und dieser auch kommerziell genutzt werden kann, sind noch einige Entwicklungen notwendig. Beispielsweise ist eine Software nötig, die das Erstellen von Ontologien und Profilen vereinfacht und es jeder Firma ermöglicht, eine solche zu formulieren. Allerdings könnten sich durch diesen Evolutionsschritt auch neue Berufsfelder entwickeln, ähnlich wie bei der Entstehung des WWW, die sich speziell mit dem Erstellen von Ontologien befassen.

Demzufolge wird es wohl noch einige Zeit dauern, bis die Entwicklung soweit voran geschritten ist, dass eine kommerzielle Verwendung von Ontologien von Industrie und Wirtschaft akzeptiert wird.

Literaturverzeichnis

[1] Achatz, M./Langhammer, T.: „Einführung in die Descrption Logics“, Universität Passau [2003]

[2] Bürki, P.: “High Precision Service Discovery“, Universität Zürich, Institut für Informatik [2003]

[3] Gruber, T. R.: “A translation approach to portable ontologies. Knowledge Acquisition”, Academic Press [1993]

[4] Haarslev, Volker/Möller, Ralf: “Racer Service Description”, http://www.sts.tu-harburg.de/~r.f.moeller/racer/ [15.06.2004]

[5] Krüger, Uwe : “Semantic Web – eine Wegbeschreibung zum WWW 2.0“, Universtität Jena [2004]

[6] Li, Lei / Horrocks, Ian: “A Software Framework For Matchmaking Based on Semantic Web Technology”, Budapest, Hungary [2003]

[7] Runggaldier, E. /Kanzian, C.: „Grundprobleme der Analytischen Ontologie“ [1998]

[8] Schadl, Bernd/ Romeis, Jens: „Ontologien für Multiagentensysteme“, Universität der Bundeswehr München, Institut Wirtschaftsinformatik [2002]