

Fakultät IV - Institut für Telekommunikationssysteme
Intelligente Netze und Management verteilter Systeme

Prof. Dr. Kurt Geihs

Michael C. Jaeger

SS 2004

Projektbericht DAML-S/OWL-S-Matcher

Eingereicht von:

Christoph Liebetruth, 188786

Berlin, 11. 09. 2004

Inhalt

1. Einleitung.....	2
2. Umstellung des Build-Environments.....	3
3. Umstellung von DAML-S auf OWL-S.....	4
3.1. Unterschiede zwischen DAML-S und OWL-S.....	4
3.2. Änderungen der Implementierung.....	6
4. Funktionserweiterung.....	8
Zusammenfassung.....	9
Verwandte Arbeiten.....	10
Literaturhinweise.....	10

1. Einleitung

Die fortschreitende Verbreitung von Web-Services entwickelt eine eigene Dynamik. Mit dem Entstehen und Vergehen von Firmen auf dem Markt ist ebenso das Entstehen und Vergehen von Web-Services verbunden. Das führt zur Entwicklung von Technologien, mit denen auf solche Ereignisse in der realen Welt dynamisch und automatisch reagiert werden kann. Eine Methode ist die semantische Beschreibung von Web-Services mittels Ontologien und einem darauf aufbauenden System zur Interpretation und Weiterverarbeitung dieser Informationen.

Eine solche Ontologie ist DAML-S. Auf Basis dieser auf XML basierten Ontologie wurde im Rahmen der Diplomarbeit von Stefan Tang an der Technischen Universität von Berlin ein Matching-Algorithmus entwickelt, der es ermöglicht, Beschreibungen von Web-Services und Anforderungen zu vergleichen und einen Grad der Übereinstimmung zu ermitteln. Der zweite Teil der Diplomarbeit bestand aus der Implementierung des Algorithmus in Form einer auf Java basierenden Anwendung.

Im Rahmen des von mir durchgeführten Projektes sollten verschiedene Umstellungen der Implementierung vorgenommen werden:

- Das Build-Environment sollte von Apache-Ant auf Apache-Maven umgestellt werden.

- Die Weiterentwicklung der Ontologie DAML-S, welche zum Zeitpunkt des Einreichens der Diplomarbeit in der Version 0.9 vorlag und welche nun in der Version 1.0 vorliegt, sollte in die vorliegende Implementierung eingepflegt werden. Mit der Weiterentwicklung von DAML-S zur Version 1.0 wurde die Ontologie in OWL-S umbenannt.

- Eine Möglichkeit von DAML-S und auch OWL-S besteht darin, die semantische Beschreibung eines Web-Service über mehrere Dateien zu verteilen. Von einer solchen Verteilung wurde in der mit der Diplomarbeit eingereichten Implementierung ausgegangen, sie ist jedoch nicht zwingend. Daher bestand der dritte Teil des Projektes in der Ergänzung des Matchers um die Funktionalität, Beschreibungen von Web-Services in Form einer einzelnen Datei zu unterstützen.

2. Umstellung des Build-Environments

Die Vorliegende Implementierung lag in einer Verzeichnisstruktur vor, die es ermöglichte, mit Hilfe des von der Apache- Foundation entwickelten Build-Tools Ant automatisiert ein Java-Archive-File zu erzeugen, welches eine vereinfachte Distribution des Matchers erlaubt.

Bei dem Build-Tool Ant handelt es sich um ein schon längere Zeit im Umlauf befindliches, auf Java basierendes Programm, welches bestehende Build-Tools wie Make und dessen Derivate ersetzen sollte, da die von diesen Tools verwendeten Make-Files nicht die benötigte Simplizität und Plattformunabhängigkeit aufwiesen. Ant nutzte als erstes Build-Tool XML für die Speicherung von Konfigurationsdateien und eliminierte damit syntaktische Schwächen von Make-Files. Durch die Abkehr von Make-Files (und damit von Shell-basierenden Skripten) und der Umstellung auf Java-basierende Erweiterungen von Ant wurde die Plattformunabhängigkeit des Build-Environments erhöht.

Das ebenfalls von der Apache-Software-Foundation entwickelte Tool Maven geht über die reine Funktionalität des Builds eines Softwarepaketes hinaus. Neben einer weiteren Vereinfachung des Build-Prozesses ist es mit Maven möglich, automatisch Softwaretests und Coverage-Checks ablaufen zu lassen, ChangeLogs und CrossReference-Dokumentationen zu erzeugen, sowie Project-WebSites automatisch zu generieren. Maven basiert dabei ebenso wie Ant auf Java und ist durch Plugins erweiterbar.

Bei der Umstellung von Ant auf Maven konnte die Verzeichnisstruktur des Projektes komplett beibehalten werden. Im Unterschied zu Ant liegen bei Maven die Packages, von denen das Projekt abhängt, nicht in einem Projekt-Unterverzeichnis „libs“ . Maven greift statt dessen auf ein globales Package-Repository zurück, dessen Position im Dateisystem oder im Netzwerk mit Hilfe einer Konfigurationsdatei spezifiziert werden kann. Durch das Einpflegen neuerer Versionen von Dependencies an einer zentralen Stelle wird die Pflege von Updates stark vereinfacht. Automatisiert ablaufende Softwaretests bei einem Rebuild eines Projektes sichern die bestehende Funktionalität auch mit neueren Versionen der Abhängigkeiten ab.

Die Maven-Konfigurationsdatei mit dem Namen „Project.xml“ liegt im Hauptverzeichnis des Projektes. Neben allgemeinen Informationen wie dem Namen des Projektes, einer kurzen Beschreibung sowie Informationen über die an der Entwicklung des Projektes beteiligten Personen enthält die Konfigurationsdatei primär Informationen über die von der Software benötigten abhängigen Pakete. Arbeitsabfolgen, die das Erzeugen des Packages betreffen, sind im Gegensatz zu Ant in dieser Datei nicht mehr zu finden, sie sind in entsprechende Plugins für Maven ausgelagert worden und stehen dem Benutzer als Funktionsbibliothek zur Verfügung. Die zur Steuerung der Plugins benötigten Parameter befinden sich in einer Datei mit dem Namen „Project-Properties“, die sich ebenfalls im Projekt-Hauptverzeichnis befindet. Damit wird der Aufwand des Builds eines Softwarepaketes auf das notwendige Minimum reduziert.

3. Umstellung des Matchers von DAML-S auf OWL-S

Im folgenden Abschnitt werden die Unterschiede zwischen DAML-S und OWL-S aufgezeigt und die damit verbundenen Änderungen der Implementierung beschrieben.

3.1. Unterschiede zwischen DAML-S und OWL-S

Die Implementierung des Matchers von Stefan Tang greift nur auf einen Teil der DAML-S-Spezifikation zurück. In der auf DAML-S basierenden Implementierung ist für das eigentliche Matching von Web-Services nur der Profile-Teil einer Web-Service-Beschreibung von Bedeutung, da hier die Funktionen des Web-Service und deren Parameter aufgelistet sind. Der Profile-Teil der Beschreibung liefert einem Agenten, der einen Web-Service sucht, Informationen darüber, was ein Web-Service tut. Neben dem Profile-Teil enthält eine DAML-S-Beschreibung eines Services noch einen Process-Teil, der noch etwas umfangreichere Informationen darüber zur Verfügung stellt, wie der Service im Detail arbeitet. Daher existiert in DAML-S eine zumindest partielle Abbildung der Parameter eines Profile's auf einen Process.

Die Beschreibung eines Parameters im Profile einer DAML-S-Beschreibung umfasst also eine Referenz auf einen Parameter im Process und neben dieser noch eine weitere Referenz auf ein sogenanntes „Konzept“ einer Ontologie.

Eine Ontologie definiert einen Namensraum. Die Elemente (Konzepte) dieses Namensraums werden in der Ontologie mittels Aussagen wie „Konzept a bedeutet dasselbe wie Konzept b“ oder „Konzept a ist eine Spezialisierung von Konzept b“ in Beziehung zueinander gesetzt. Die Benennung von Funktionsparametern legt für einen Benutzer eines Web-Service zwar eine semantische Bedeutung nahe, für eine maschinelle Auswertung sind sie jedoch nicht ausreichend, da es sich für einen Computer nur um beliebige Strings handelt, auf deren Basis kein Matching möglich ist. Erst die Zuweisung eines Konzeptes zu einem Parameter, das „Wissen“ über die Beziehungen von Konzepten untereinander und die Verwendung der gleichen Ontologie für verschiedene Web-Services des gleichen Aufgabenbereiches sorgt für eine Vergleichbarkeit und ein darauf aufbauendes automatisiertes, semantisches Matching.

In einer DAML-S-Beschreibung wird auf das dem Parameter entsprechende Konzept aus sowohl aus dem Profile als auch aus dem Process heraus verwiesen.

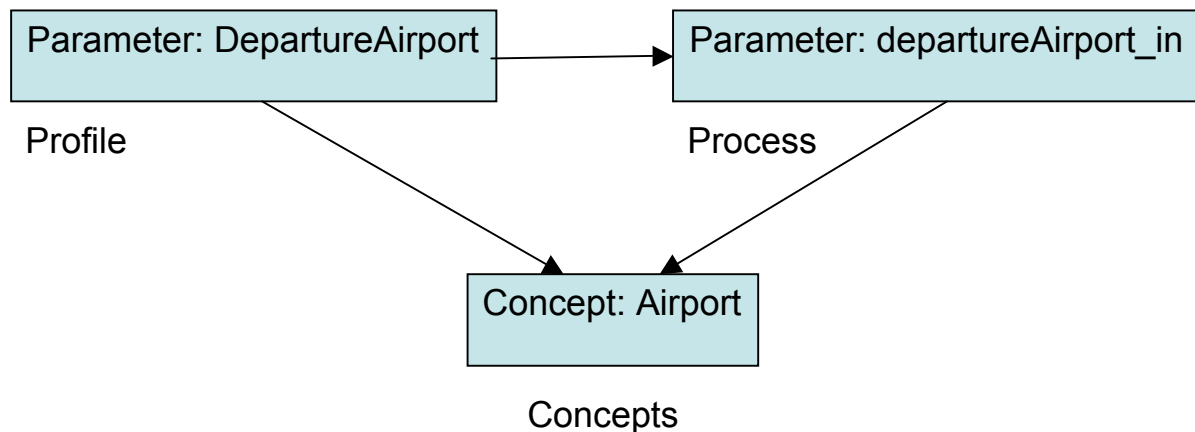


Bild 1: Parameter-Referenzierung mit DAML-S

Die Verweise auf das Konzept sind, wie aus der Abbildung ersichtlich, redundant. In der Aufhebung dieser Redundanz liegt einer der Unterschiede zwischen DAML-S und OWL-S. In OWL-S wird nur noch aus dem Process heraus auf das hinter einem Parameter liegende Konzept verwiesen.

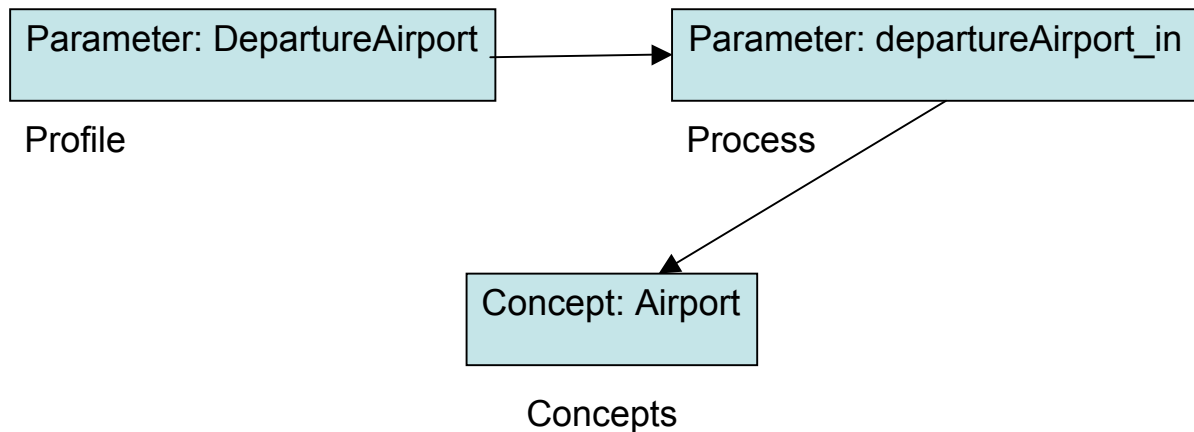


Bild 2: Parameter-Referenzierung mit OWL-S

Neben dieser Aufhebung der Redundanz wurden bei der Weiterentwicklung von DAML-S 0.9 zu OWL-S 1.0 Informationen, die Kontaktdaten der einen Service zur Verfügung stellenden Organisation beinhalten, aus dem XML-Profile-Namespace in einen eigenen „Actor“-Namespace verschoben worden. Die Deklaration der Actor-Klassen und ihrer Properties erfolgt bei OWL 1.0 in einer eigenen Datei ActorDefault.owl.

Ein weiterer Unterschied findet sich bei der Definition von Quality-Of-Service-Parametern und Subtypen der ServiceCategory-Klasse. Die Definition erfolgt bei OWL 1.0 ebenfalls nicht mehr im Profile-NameSpace, sondern in einem eigenen „ProfileAdditionalParameters“-NameSpace.

3.2. Änderungen der Implementierung

Der von Stefan Tang im Rahmen der Diplomarbeit entwickelte Matcher basiert auf dem Komponente DAMLJessKB, einem DAML-Reasoner auf der Basis von JESS (Java Expert System Shell). Dieses Modul übernimmt das Einlesen von DAML-S-Web-Service-Beschreibungen und den referenzierten Dateien in das Expert-System. Das Expert-System kann dann abgefragt werden, wobei eine begrenzte Zahl von Standard-Abfragen bereits von DAMLJessKB zur Verfügung gestellt wurden.

Die zu vergleichenden Beschreibungen werden in das Expert-System eingelesen und es wird anschließend für jede Funktion des einen Services und jeden Parameter desselben versucht, eine äquivalente Funktion des anderen Service zu finden. Das Expert-System hilft dabei, die Konzepte der Parameter zu vergleichen, in hierarchische Beziehung zueinander zu setzen und den Grad an Übereinstimmung zu ermitteln.

Bei der Umstellung des Matchers von DAML-S auf OWL-S wurde ein Austausch der DAMLJessKB-Komponente gegen die OWLJessKB-Komponente notwendig, die eine ähnliche Funktionalität für OWL-S-Beschreibungen liefert. Dieser Austausch zog jedoch weitere Änderungen nach sich.

Die JessKB-Komponenten stellen einen Wrapper um die Jess-Knowledge Base dar und ermöglichen es den Benutzern der Komponenten, die Knowledge Base abzufragen. Im Unterschied zu OWLJessKB gab es bei DAMLJessKB Funktionen für den einfachen Zugriff auf die Knowledge Base (SimpleSubjectQueries), sowie eine Reihe von bereits in die Knowledge Base geladenen, vordefinierten Standardabfragen.

Bei der Weiterentwicklung von DAMLJessKB zu OWLJessKB wurde die Funktion für einfache Abfragen sowie die Standardabfragen aus OWLJessKB entfernt, so daß diese, soweit sie vom Matcher genutzt wurden, in die Implementierung des Matchers übernommen werden mußten. Die Reasoner-Klasse des Matchers, welche mit der Jess-Knowledge-Base interagiert, ist daher um die aus der DAMLJessKB-Komponente entfernte Funktionalität für SimpleSubjectQueries gewachsen. Bei der Initialisierung der Jess-Knowledge-Base im Konstruktor der Reasoner-Klasse werden außerdem die fehlenden, sowie eine Menge zusätzlicher Abfragen in das Expert System geladen.

Die Reasoner-Klasse hat damit an Bedeutung als Fassade für die Interaktion mit dem Expert System durch Zunahme des Funktionsumfangs gewonnen. Für alle Abfragen, die für den Matching-Algorithmus, wie auch das Parsen der Service-Klassenstruktur relevant sind, existieren öffentliche Funktionen, so daß von der Internen Syntax der in der Knowledge Base hinterlegten Fakten abstrahiert wird.

Weiterhin hat sich mit dem Wechsel von DAMLJessKB zu OWLJessKB die Syntax der in der Knowledge Base abgelegten Fakten geändert. Diese Änderung zog, da die Syntax der Abfragen in Beziehung zur Syntax der Fakten steht, eine Änderung aller bislang genutzten Abfragen nach sich.

Da die XML-basierte Beschreibung der Services in sehr hohem Maße auf XML-Namespaces basiert, und diese den Namen der Ontologien sowie deren Version enthielten, war auch hier eine durchgehende Anpassung vonnöten.

4. Funktionserweiterung

Bei der Umstellung des Matchers wurde die Funktionalität mit Hilfe von verschiedenen OWL-S-Beschreibungen überprüft. Bei den zum Zeitpunkt der Abgabe der Diplomarbeit von Stefan Tang auffindbaren Beschreibungen waren die Informationen über mehrere Dateien verteilt, und jede dieser Dateien beinhaltete einen der in der DAML-S-Spezifikation definierten Anteile, nämlich Service-, Profile-, Process-, und Grounding-Anteil. Das suggerierte eine zwingende Aufteilung einer Beschreibung in diese Dateien, die so von der Spezifikation nicht gefordert, im Matcher jedoch ausgenutzt wurde. Der Einfachheit halber wurden außerdem die Inhalte der Dateien in der vorliegenden Implementierung mit Hilfe des XML-Parsers ausgelesen.

Da sich gezeigt hat, daß von einer solchen Aufteilung nicht ausgegangen werden kann, und obendrein die Struktur der XML's keinem statischen Schema folgt, mußte die Implementierung dahingehend abgeändert werden, daß die für den Matcher relevanten Informationen ausschließlich aus dem Expert-System ausgelesen werden. Das machte eine signifikante Erweiterung der Reasoner-Klasse, welche die JessKB-Instanz hält und mit dieser interagiert, notwendig.

In der angepaßten Version des Matchers wird ausschließlich die OWL-Datei, die die ServiceID eines Web-Services enthält, mit Hilfe des XML-Parsers eingelesen, um genau diese zu ermitteln. Anschließend wird die Datei der OWLJessKB-Komponente übergeben, welche den Inhalt der Datei der Knowledge Base hinzufügt. Referenziert die OWL-Datei über Import-Statements weitere OWL-Dateien, die Teile der OWL-S-Beschreibung enthalten, so werden diese von der OWLJessKB-Komponente automatisch geladen. Damit wurde von der Aufteilung der Informationen über eine oder mehrere Dateien, sowie von deren syntaktischem Aufbau abstrahiert.

Die ServiceID dient dem Matcher als Einstiegspunkt. In der bestehenden Implementierung wurde von einer festen Position der Service-ID innerhalb einer OWL-Datei ausgegangen. Da aber auch sie keine feste Position hat, muß sie mit

Hilfe eines XPath-Ausdrucks ermittelt werden. Diese Funktionserweiterung zog einen Austausch der JDOM-Komponente und der Einbindung des SAX-Parsers nach sich. Weder der Austausch der JessKB-Komponenten noch die Funktionserweiterung haben zu einer signifikanten Änderung der Klassenstruktur geführt. Die Reasoner-Klasse wurde erweitert und die Parser-Klasse umgeschrieben, so daß das Einlesen der OWL-Files in die bereits bestehende Klassenstruktur ermöglicht wurde. Der eigentliche Matching-Algorithmus baut auf der bestehenden Klassenstruktur auf und mußte daher nicht geändert werden.

Zusammenfassung

Die grundlegende Struktur der semantischen Beschreibungen von Web-Services, nämlich die Einteilung in Service-, Process-, Profile- und Groundinganteil blieb beim Versionswechsel von DAML 0.9 zu OWL 1.0 erhalten. Der strukturelle Aufbau der Anteile ist derzeit jedoch noch teilweise erheblichen Änderungen unterworfen.

Der Versionswechsel wurde von den Entwicklern der DAML/OWL-JessKB-Komponenten dazu genutzt, das Reasoning mit Hilfe des Expert Systems zu ändern und zu optimieren. Es ist aber davon auszugehen, daß dieser Teil in nächster Zeit keine größeren Änderungen erfahren wird.

Die Entwicklung von OWL-S ist indes noch nicht abgeschlossen. Zum Ende dieser Ausarbeitung befindet sich OWL-S bereits im Beta-Stadium der Version 1.1.

Neben der reinen Definition des Aufbaus von OWL-S-Beschreibungen ist für eine weite Verbreitung dieser Technologie noch viel Arbeit vonnöten: Um ein effektives Matching zu ermöglichen und auf die Bedeutung von Funktionen von Services schließen zu können, müssen diese mit Worten innerhalb von gemeinsamen Sprachkontexten, von gemeinsamen Konzepten beschrieben werden. Die Definition von Konzepten steckt jedoch noch in der Anfangsphase der Entwicklung und die Verbreitung und Akzeptanz wird noch einen längeren Zeitraum in Anspruch nehmen.

Verwandte Arbeiten

[1] Carnegie Mellon University: Semantic Matching of Web-Services

<http://www-2.cs.cmu.edu/~softagents/daml.html>

[2] Web Service Composer

<http://www.mindswap.org/~evren/composer>

Literaturverzeichnis

[1] Diplomarbeit Stefan Tang

http://ivs.tu-berlin.de/Jaeger/damlsmatcher/resources/thesis_stefang.pdf

[2] DAML Services

<http://www.daml.org/services/daml-s/0.9>

[3] OWL Services

<http://www.daml.org/services/owl-s/1.0>

[4] OWLJessKB

<http://edge.cs.drexel.edu/assemblies/software/owljesskb>

[5] Java Expert System Shell

<http://herzberg.ca.sandia.gov/jess>