Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Projekt-Ausarbeitung

Piotr Wendt

Semantic Web Service Suche

Betreuender Prüfer: Prof. Dr. Kai von Luck

Fakultät Technik und Informatik
Studiendepartment Informatik

Piotr Wendt

Thema der Projekt-Ausarbeitung

Semantic Web Service Suche

Stichworte

Semantic Web, Web Services, Semantic Web Services, Suche

Kurzzusammenfassung

Dieser Arbeit stellt eine Beschreibung der in Rahmen des Ferienklubprojektes durchgeführten Arbeiten. Eine der Teilaufgabe war in diesem eine effizientere Suche von Diensten zu ermöglichen. Hierzu sollte diese durch die aus dem Semantic Web stammende Ansätze semantisch erweitert werden, was zu Effizienzsteigerung der Suche führen sollte. Dabei wurden verschiedene Verfahren aus einigen bestehenden Forschungsprojekten evaluiert und auf ihre Einsetzbarkeit geprüft. Neben der Beschreibung der einzelnen Methoden wird hier die Architektur eines möglichen Lösungsansatzes beschrieben. Im letzen Teil der Ausarbeitung folgt dann eine Bewertung der hierbei gewonnenen Erkenntnisse und Kritikpunkte. Zum Schluss wird ein möglicher Ausblick gegeben, wie die hier angefangenen Arbeiten weiter verfolgt werden könnten.

Inhaltsverzeichnis

1 Einleitung **1**

2 Hintergrund **2**
2.1 OWL-S 2
2.2 Protege OWL-S Editor 3
2.3 OWL-S IDE 3
2.4 OWLS VM 4
2.5 jUDDI 4
2.6 OWL-S Matchmaker 5
2.7 Weitere Werkzeuge 5

3 Entwurf **6**
3.1 Gesammt Architektur 6
3.2 Mapping Modul 7
3.3 Merging Modul 8
3.4 Transfomationsmodul 8
3.5 MatchAgent 8
3.6 Persistenzschicht 9

4 Realisierung **9**

5 Fazit **10**
5.1 Zusammenfassung 10
5.2 Schlussfolgerungen und Kritik 10
5.3 Ausblick 12

1 Einleitung

In Rahmen des, an der HAW angebotenen, Masterstudiums fand eine Veranstaltung Projekt statt, in der ein fiktives Szenario eines Ferienklubs aufgestellt wurde. Dieses stellte die Grundlage für weitere Betrachtung und Analyse der in solchem Umfeld entstehenden IT relevanten Problemstellungen. Diese Problemstellungen sollten dann nach Möglichkeit von einzelnen Projektgruppen analysiert und weit gehend gelöst werden, wobei das Durchdringen der Materie und Gewinnen neuer Erkenntnisse im Vordergrund stand.

Es bildeten sich mehrere Arbeitsgruppen, eine darunter beschäftigte sich mit der Semantic Web Technologie. Diese hatte die Aufgabe Semantic Web Technik im Umfeld des Ferienklubs einzusetzen, bzw. deren Einsetzbarkeit zu evaluieren. Zum anderen sollte eine prototypische Implementierung der sich hierbei herauskristallisierten Ideen durchgeführt werden.

Als Ergebnis der vorher in (Wendt, 2005a) und (Wendt, 2005b) durchgeführten Untersuchungen ergab sich eine Architektur, durch die der Ferienklub Mitarbeiter sowie der Gast bei der Planung der Freizeit unterstützt werden sollte. Diese soll die beiden involvierten Gruppen bei der Auswahl der Freizeitaktivitäten unterstützen, indem es die Kommunikation mit den entsprechenden Dienstanbietern vereinfacht. Zusammengefasst soll diese die Heterogenität der einzelnen, Außerhalb des Ferienklubs existierenden, Dienste verbergen und somit eine nahtlose Kommunikation ermöglichen. Um dieses zu erreichen, wurden von der Semantic Web Gruppe folgende resultierenden Ziele definiert:

- Um die vorhandenen syntaktischen Strukturen sowie die neu definierten semantischen Daten gemeinsam verwenden zu können, muss eine geeignete Methode gefunden werden, die diese auf einander abbildet. Zudem sind Mechanismen notwendig, die hierbei verarbeitenden Komplexitäten intelligent verteilt und entschärft.

- Die verteilten Dienste unterscheiden sich sehr in ihren Aufbau sowie deren intern verwendete fachlicher Terminologie. Diese kann mit Hilfe von Ontologien formal spezifiziert werden und semantisch beschrieben werden. Die Heterogenität der einzelnen Dienste kann dann durch Bildung einer gemeinsamen Fachsprache verborgen werden. Dabei müssen zwei unterschiedliche terminologische Sichten durch geeignete Strategien zusammengeführt werden.

- Die neuen, durch Semantik angereicherten, Strukturen müssen auf die bereits vorhandenen abgebildet werden. Dieses ist auch in der umgekehrten Richtung notwendig. Dabei werden neue Strukturen auf bereits bekannte transformiert und damit den bestehenden Technologien zur Verfügung gestellt. Durch diese Transformation sollen also Daten in verschiedene, unterschiedliche Formate transformiert und somit jeweiligen Anwendungen zur Verfügung gestellt werden.

- In dieser Architektur Spielen Dienste eine Schlüsselrolle, solange einige Vorgaben erfühlt sind. Eine dieser Vorgaben ist, dass diese auffindbar gemacht werden. Hierfür muss eine geeignete Möglichkeit geschaffen werden, durch die, Dienste genauer beschrieben werden können. Dadurch sollen implizite Zusammenhänge zwischen den Diensten analysierte werden können. Dieses, der Suche zugänglich gemachtes, implizites Wissen soll dann eine effizientere Suche ermöglichen.

Diese Arbeit beschreibt die in Rahmen des Studienprojektes durchgeführten Arbeiten, welche speziell die Suche der semantisch beschriebenen Dienste betrachtet. Dabei wird in Abschnitt 2 die für die Lösungsfindung notwendigen Werkzeuge und Techniken kurz identifiziert und beschrieben. Als nächstes wird in Abschnitt 3 der Entwurf der möglichen Plattform der verbesserten Suche vorgestellt. In Abschnitt 4 sollen die prototypisch erstellten Anwendungen vorgestellt werden. Zum Schluss im Abschnitt 5 soll dann ein Fazit gezogen und ein Ausblick gewagt werden.

2 Hintergrund

2.1 OWL-S

Bei der Suche der bestgeeigneten Dienste sollten möglichst viele Eigenschaften dieser betrachtet werden. Damit wird die Genauigkeit der daraus resultierenden Treffer erhöht und die Ergebnismenge erheblich reduziert. Die so gewonnene, auf wesentliche Merkmale eingeschränkte Ergebnismenge kann dann gezielter und effizienter durchsucht werden, was zu einer Reduzierung der Komplexität führt.

Um diese zur Beginn genannte akkurate Beschreibung der Dienste zu ermöglichen, werden formal spezifizierte Beschreibungssprachen zu Hilfe genommen. Diese basieren auf Ontologien, die ein formal spezifiziertes Vokabular darstellen, wodurch neue Möglichkeiten der maschinellen Verarbeitung geöffnet werden. Eine solche Beschreibungssprache ist OWL-S. Diese fungiert im diesem Projekt als Basisontologie und Beschreibungsgrundlage der hier zum Einsatz kommenden Dienste.

OWL-S (OWL-S, 2005) teilt die Beschreibung der Dienste in drei Kategorien, Service Profile, Process Model und Grounding. *Service Profile* beschreibt die Diensteigenschaften und diverse andere Charakteristika, welche bei einer gezielten Suche berücksichtigt werden [1]. Das *Process Model* beschreibt die technischen Einzelheiten des Dienstes. Daneben beschreibt dieses formalisiert deren interne Funktionsweise. Aus dieser Beschreibung kann man Schlussfolgern, wie eine Kommunikation mit diesem durchgeführt werden kann. Die dritte Komponente dieser Beschreibungssprache ist *Service Grounding*. Dieses spezifiziert

[1] In dem Service Profile können z. B. die vier grundlegenden Eigenschaften wie Eingabe, Ausgabe, Vorbedingungen und Auswirkungen eines Dienstes beschrieben werden.

die Abbildung der in Process Model deklarierten abstrakten Interaktionen und konkretisiert diese in Form von in "WSDL"definierten Nachrichten. Genauere Beschreibung von OWL-S und die Entscheidungsgrundlage für deren Auswahl findet man in (Wendt, 2005a) und (Wendt, 2005b).

2.2 Protege OWL-S Editor

Die zuvor beschriebene Beschreibungssprache OWL-S basiert auf OWL (W3C, 2004) und stellt selber eine spezialisierte Upper Ontologie dar, die eine formalisierte Richtlinie zur Beschreibung der Dienste einführt. Neben den durch diese eingeführten formalisierten Charakteristiken, werden weitere hinzugefügt und weiter durch weitere Domänen spezifische Ontologien erweitert. Hierdurch wird ein spezialisiertes Vokabular durch höherwertiges abstrahiert und somit relativiert.

Diese Verschachtelung der verschiedenen Ontologien führt zu Verlust der Übersichtlichkeit, was eine herkömmliche Editierung erheblich erschwert. Konsequenterweise sind hierfür Entwicklungswerkzeuge notwendig, welche diese Hürde etwas entschärfen und eine Entwicklung der Semantic Web Services (SWS) erleichtern sollen.

Protege Stanford (2005) stellt ein solches Werkzeug dar. Dieses erlaubt die Editierung von verschiedenen Ontologien, deren Validierung und Zusammensetzung. Daneben erlaubt es einfache Abfragen dort entwickelten Ontologien auf implizite Zusammenhänge[2]. Des Weiteren ist Protege durch die Plugin Architektur erweiterbar, sodass die Anbindung von OWL-S relevanten Werkzeugen wie *OWL-S Editor* (Semwebcentral.org, 2005) möglich ist. Dieser betrachtet speziell die Einzelheiten der OWL-S Ontologie und erlaubt die Editierung der drei OWL-S Hauptkomponenten. Daneben stellt *Graphviz* (graphviz.org, 2005) Möglichkeiten der grafischen Editierung der internen Prozessketten der Dienste bereit.

2.3 OWL-S IDE

Eine alternative Entwicklungsumgebung stellt die OWL-S IDE ((Srinivasan u. a., 2006), (Srinivasan u. a., 2005)) dar. Diese wurde dazu konzipiert den Entwickler von SWS bei der Entwicklung, Deployment und Prozessbeschreibung zu unterstützen. Dieses kombiniert extern entwickelte Werkzeuge zusammen zur einer Eclipse basierten Entwicklungssuite. Diese unterstützt den Entwickler über den kompletten Entwicklungsprozess hinweg. Nach (Srinivasan u. a., 2006) ermöglicht diese die Entwicklung der OWL-S basierter Beschreibung, Publizierung so beschriebenen Diensten, deren Auffindung und Ausführung.

Eine Teilkomponente von OWL-S IDE ist CODE (CMU's OWL-S Development Environment), welches als Eclipse Plugin zur Verfügung steht. Diese Entwicklungsumgebung bietet die Möglichkeit an, den *Code-Driven* sowie den *Model-Driven* Ansatz zu verfolgen.

[2] Hierfür wird intern JENA verwendet

Beim ersten Ansatz wird aus einem Javaprogramm mit Hilfe von JAVA2OWL-S (semwebcentral.org, 2005a) Werkzeug eine rohe OWL-S Datei erzeugt. Diese muss dann mit Hilfe eines OWL-S Editors, welcher ebenfalls Bestandteil der OWL-S IDE ist, mit Domänen abhängige Semantik angereichert werden.

Der Model-Driven Ansatz geht den umgekehrten Weg. Diese ermöglicht die Generierung von konkreten Javaprogrammen aus OWL-S Spezifikationen. Ist die Beschreibung semantisch angereichert, bietet CODE ebenfalls eine Möglichkeit an diese auf syntaktische und logische Fehler zu validieren. Zum Schluss unterstützt CODE ebenfalls die Testphase, indem diese das resultierende Programm auf einer integrierten Virtuellen Maschine ausführt.

Bei dem Code-Driven Ansatz werden unter anderen auch zwei andere Hilfswerkzeuge genutzt, das WSDL2OWL-S (semwebcentral.org, 2005b) und OWL-S2UDDI (semwebcentral.org, 2005b). WSDL2OWL-S generiert aus einer in WSDL Datei spezifizierten Beschreibung eines Diensten, eine OWL-S Datei, welche durch Nachbearbeitung mit Semantik angereichert werden kann. Das OWL-S2UDDI ist ein Werkzeug, welches dann die OWL-S Beschreibung auf die, in der UDDI definierten TModels abbildet.

2.4 OWLS VM

Die OWL-S Virtuelle Maschine (Paolucci u. a., 2003) ist eine Laufzeitumgebung, welche Kontrolle über die Zugriffe auf OWL-S Dateien, dort referenzierten Ontologien und den Web Services hält. Diese Virtuelle Maschine benutzt das OWL-S Grounding um konkrete WSDL Operation aus dem abstrakten Anfragen zu generieren, welche zwischen einem Anbieter und dem Anfragendem ausgetauscht werden. Hierzu wird eine Inferenzmaschine zu Hilfe genommen, um aus der Beschreibung und den zur Verfügung gestellten Ontologien auf die Dienstimplementierung schließen zu können. OWL-S VM unterstützt eine Clientanwendung indem es Vorgaben, aus dem zugrunde liegendem Prozess Model, zur Implementierung von Operationen liefert.

2.5 jUDDI

Zurzeit besteht die Suche von Diensten darin, Registrierungsstellen, in den diese registriert wurden, auf deren Existenz zu befragen. jUDDI Apache.org (2005) ist eine Ausprägung solcher Registrierungsstelle. Diese beschreibt die dort registrierten Dienste mit Hilfe von so genannten TModels. TModel stellt ein Referenzsystem dar, in dem Metadaten über Dienste abgelegt werden können. Deren Aufgabe liegt zum einem darin, den Servicetyp und die Nutzungsweise eines Dienstes zu identifizieren. Zum anderen bietet diese die Möglichkeit an die Dienstmerkmale mit Hilfe von Taxonomien zu annotieren. Auf der Basis dieses Referenzsystems unterstützt jUDDI eine Schlagwortsuche, um die dort residierenden Dienste wieder finden zu können. jUDDI ist eine reine Java Web Anwendung, die auf dem Jakarta Tomcat aufsetzt. Daneben benötigt diese eine externe Datenbank.

2.6 OWL-S Matchmaker

Wie bereits verdeutlicht, basiert die Beschreibung der in jUDDI abgelegter Dienste auf Merkmalsattributen (TModels), sowie intern verwendeten Taxonomien. Leider betrachtet die dort realisierte Schlagwortsuche keine impliziten Zusammenhänge zwischen den einzelnen Diensten. Zum anderen werden ebenfalls die semantischen Zusammenhänge zwischen den beiden Sichten, der Anbietersicht und der Anfragendensicht, nicht berücksichtigt.

Mit Hilfe einer Matchingmaschine sollte die Diskrepanz zwischen den beiden Kommunikationspartnern relativiert werden. Dazu untersucht diese die Beschreibungen des Anbieters und des Anfragenden mit Zuhilfenahme von explizitem Wissen, um somit diese entsprechend zu identifizieren und die Diskrepanz aufzudecken und entsprechend zu beheben. OWL-S Matchmaker stellt eine solche Komponente dar.

In (Paolucci u. a., 2002b), (Paolucci u. a., 2002a) und (Srinivasan u. a., 2004) wird der OWL-S Matchmaker Konzept vorgestellt der oben beschriebenen Problemstellungen beseitigen soll. Dabei wird in (Paolucci u. a., 2002b) die formale Grundlage für Matchingalgorithmen gegeben, sowie deren prototypische Implementierung diskutiert.

2.7 Weitere Werkzeuge

Außer den bereits genannten Werkzeugen sind für die Realisierung weitere notwendig. Axis (Avar u. a., 2005) stellt einen Web Service Container dar, welcher für die Kommunikation erforderliche Protokolle implementiert. Diese wird ebenfalls wie jUDDI in den Jakarta Tomcat installiert.

RACER (RACER-Systems, 2005) stellt einer Implementierung einer Inferenz Maschine dar. Diese ermöglicht aus dem gegebenen Wissen Rückschlüsse zu ziehen und damit auf die, in Daten verborgenen, Zusammenhänge zu resultieren. Dieser kann dann eine Bewertung der Annahmen durchführen, wie weit ein oder mehrere Dienste der angeforderten Anforderungen genügen.

Im weiteren Verlauf dieser Arbeit würden zusätzlich noch das WSDL2OWL-S (semwebcentral.org, 2005b) und JAVA2WSDL (semwebcentral.org, 2005a) benutzt. WSDL2OWL-S generiert aus einer WSDL Datei für jeder WSDL Operation einen OWL-S Process Model. Daneben wird das komplette OWL-S Grounding und Teile des Profiles generiert. Um aus einer Java Datei WSDL zu generieren wurde hierzu aus Axis das Programm Java2WSDL zu Hilfe genommen.

3 Entwurf

3.1 Gesammt Architektur

In (Wendt, 2005a) und (Wendt, 2005b) beschriebenes Anwendungsszenario wird eine zentrale Komponente identifiziert. Diese fungiert als Mediator zwischen den in Ferienklub und Außerhalb existierenden Diensten. Diese soll die Heterogenität nicht nur auf der syntaktischen sondern auch auf der semantischen Ebene transparent für die Clientanwendungen machen. Hieraus resultierende Vorteile werden bei der Entwicklung der Clients ersichtlich, da diese die Heterogenität nicht betrachten müssen.

Eine solche Mediationskomponente muss folgende Aufgaben verrichten:

- Dienste finden und identifizieren können
- Diese muss im Stande sein mit Diensten zu kommunizieren und sich somit an syntaktische Strukturen anpassen können
- Außerdem muss diese die Dienste interpretieren können um die Zusammenhänge zwischen unterschiedlichen Diensten erkennen und bewerten können

Auf der Grundlage der eben identifizierten Aufgaben können jeweilige Zuständigkeiten auf einzelne Module unterteilen werden.

- Mapping Modul: Hat die Aufgabe die semantischen Informationen auf bereits vorhandene Strukturen, in diesem Fall UDDI TModels, abzubilden.
- Merging Modul: Hat die Aufgabe unterschiedliche Begriffsmodelle zur einen in sich konsistentem Begriffsmodell zu überführen.
- Transformationsmodul: Dieser unterstützt das Mapping Modul, in dem es die unterschiedlichen Formate sowie unterschiedlichen Datentypen konvertiert.
- Persistenzschicht: Diese speichert die Abbildungen zwischen den in der UDDI vorhanden TModels und den semantischen Informationen.
- MatchAgent: Seine Hauptaufgabe ist die vorhandenen Informationen in Form von Ontologien (Domänenontologien) auf semantische Zusammenhänge zu überprüfen und zu Bewerten.

Eine grobe Skizze dieser Architektur wird in 1 dargestellt.

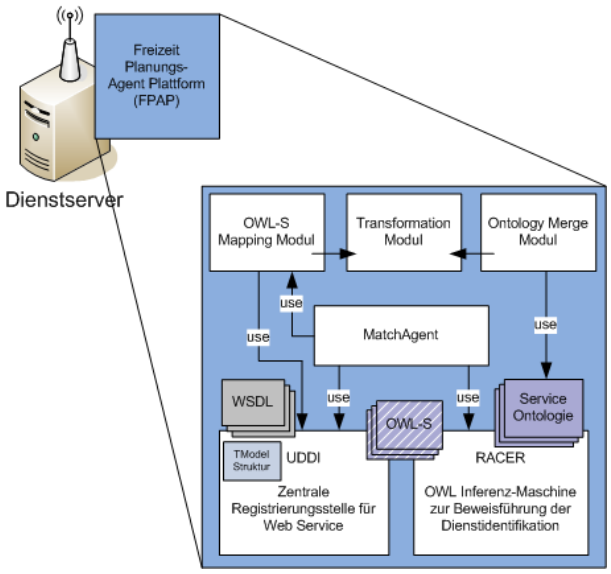

Abbildung 1: Architekturskizze der Mediatorkomponente.

3.2 Mapping Modul

Mapping Modul kapselt die notwendige Funktionalität zur Abbildung von semantischen Strukturen auf syntaktische UDDI TModel Struktur. Der umgekehrte Weg wird ebenfalls unterstützt. Durch diesen werden die beiden semantische und syntaktische Ebenen in Beziehung zu einander gesetzt. Dieses ermöglicht den parallelen Betrieb der beiden Ansätze, was der Kompatibilität an die bereits bestehender UDDI Infrastruktur zugute kommt.

Die durchzuführenden Aufgaben werden in zwei Phasen unterteilt, die Dienstpublizierung und die Dienstsuche. Um den Unterschied deutlich zu mache wird an dieser Stelle die Funktionsweise separat in den jeweiligen Phasen beschrieben, wobei die Suchphase in den MatchAgenten Modul beschrieben wird.

Bei der Dienstpublizierung wird von dem Anbieter des Dienstes eine WSDL, eine OWL-S Datei sowie die Domänenontologie übermittelt. Daneben kann dieser die konventionellen Möglichkeiten über die Standardschnittstelle der UDDI nutzen, um den Dienst zu beschreiben. Nach Beendigung der Dienstpublizierung von dem Dienstanbieter, startet das Mapping Modul und betrachtet die in den TModels abgespeicherten Daten, die WSDL und die OWL-S Datei. Dabei veranlasst dieser das Merging Modul eine gemeinsame Wissensbasis zu berechnen. Basierend auf diesem Wissen, versucht das Mapping Modul die in der OWL-S abgelegte semantische Informationen mit den entsprechenden TModels zu verbinden ggf. legt neue an. Durch diese Aufbereitung soll die in den Ontologien enthaltene Komplexität entschärft werden, was in der zweiten Phase, der Dienstsuche zugute kommen kann.

3.3 Merging Modul

Wie zuvor erwähnt werden in dieser Architektur zur Nivellierung der Heterogenität der unterschiedlichen Dienste drei Ontologien benutzt. Jeder Dienstanbieter beschreibt sein Dienst mit Hilfe einer Upper Ontologie, in diesem Fall OWL-S und reichert diese mit einer Domänen Ontologie an. Zum anderen hält die zentrale Kommunikationskomponente eine Ferienklubontologie bereit, in der das ferienklubrelevante Wissen formal spezifiziert ist. Da das OWL-S gleichzeitig eine standardisierte Ontologie darstellt, können diese hierfür relevanten Aspekte in die Implementierung des Merging Moduls einfließe (zwar wird hierdurch eine Abhängigkeit zu diesem Standard geschaffen, auf der anderen Seite ermöglicht dieses eine Optimierung in der Verarbeitung solcher Ontologie, was einen Performance Zuwachst darstellen kann). Aus dieser Gegebenheit müssen nur die beiden so genannten Domänenontologien relativiert werden. Hierzu werden diese durch geeignete Heuristiken transformiert und unter Beachtung der Konsistenzregeln auf einen gemeinsamen Nenner gebracht. Das Transformationsverfahren delegiert das Merge Modul an das Transformationsmodul.

Daneben greift das Merge Modul auf eine Inferenz Maschine. Diese dient zur Beginn der Merge Phase als eine Wissensbasis auf der Annahmen überprüft werden können, wie die beiden Ontologien zusammengeführt werden können. Würde eine Zusammenführung durchgeführt, veranlasst das Merging Modul die Inferenzmaschine eine Prüfung der Konsistenz der resultierenden Ontologie durchzuführen. Nach der erfolgreichen Überprüfung der resultierenden Ontologie, wird diese persistent gemacht. Die beiden ursprünglichen Ontologien bleiben unverändert. Dieses ist notwendig, damit durch fortlaufende Veränderung der Ontologie keine Verfälschung bzw. Verschlechterung der Zusammenhänge verursacht wird.

3.4 Transfomationsmodul

In den vorherigen Abschnitten wurden die Transformationsaufgaben motiviert. Unklar ist ob das Transformationsmodul das, für die Transformation notwendiges, Wissen von den Mapping- und Merging Modulen geliefert bekommen sollte, oder ob das Transformationsmodul selber dieses Wissen von den entsprechenden Komponenten holen soll.

3.5 MatchAgent

Der MatchAgent stellt in dieser Architektur eine zentrale Komponente dar, welche bei der Dienstsuche, aber auch in der Publizierungsphase beansprucht wird. Während der Publizierungsphase wird eine Abbildung der semantischen Daten auf die syntaktischen durchgeführt (Normalisierung). Gleichzeitig werden aber diese auch einer Bewertung unterzogen. Dabei wird eine Ähnlichkeitsuntersuchung der einzelnen Ontologiekonzepte der Dienstbeschreibung mit der Ferienklubontologie durchgeführt. Durch geeignete Matchingalgorithmen kann

an dieser Stelle ein Matchinggrad errechnet werden und in den entsprechenden TModels abgelegt werden.

In der Suchphase muss die Anfrage so weit aufbereitet werden und mit der bestehenden Wissensbasis abgeglichen werden. Dabei werden zuerst die in der Anfrageontologie beschriebenen Grundeigenschaften auf die Übereinstimmung mit den UDDI TModels untersucht. Daraus resultierende Dienstmenge wird dann durch das MatchAgent Modul weiter auf die Übereinstimmung untersucht. Hierzu werden die bereits errechneten Matchinggrade berücksichtigt.

3.6 Persistenzschicht

Bei dem Vorgang der Abbildung der semantischen Daten auf syntaktische, wird eine Verteilung der Daten durchgeführt. Dabei müssen die Relationen in der Datenbank abgespeichert werden. Diese sind notwendig, um aus den in der UDDI abgelegten TModels wieder die semantischen Informationen laden zu können. Dieser Eingriff erlaubt das parallele Betreiben von zwei Konzepten. Zum einen kann die einfache UDDI Funktionalität weiterhin angeboten werden. Zum anderen hat man die Möglichkeit auf die semantisch berücksichtigende Registrierungsstelle zuzugreifen. Daneben kann in die Datenbank neben den Relationen ein Matchinggrad abgelegt werden.

4 Realisierung

Wie bereits erwähnt, verfolgte diese Arbeit weniger eine Neuimplementierung von Ansätzen als Ziel. Im Vordergrund stand die Wiederverwendung bereits vorhandener Lösungen. Hierzu wurde RACER als Inferenz Maschine benutzt. Eine Java Implementierung des UDDI Konzeptes jUDDI. Des Weiteren wurde für die Implementierung der prototypischen Dienste Eclipse benutzt. Es wurden drei rudimentäre Dienste *CarReservation*, *CarSelection* und *VehicleSearch* ansatzweise implementiert.

An dieser Stelle wurde versucht die so definierten Klassen weiter mit der CODE Entwicklungsumgebung zu vollwertigen Diensten zu implementieren. Durch einige Fehlschläge wurde der manuelle Ansatz gewählt, indem aus dem Java Dateien WSDL Beschreibungen generiert wurden.

Die so gewonnenen WSDL Dateien dienten dann als Grundlage für den oben beschriebenen WSDL2OWL-S Konverter, der OWL-S Dienstbeschreibungen generiert hat. So generierten OWL-S Dateien enthielten keine bzw. nur ansatzweise semantische Daten. Diese müssten nachträglich hinzugefügt werden. Hierzu müssten Domänenontologien bereitgestellt werden, auf deren Basis diese Beschreibungen dann ansetzten. Hierfür würde das Protege benutzt, welches zurzeit das gängige Entwicklungswerkzeug für Ontologien darstellt.

An dieser Stelle folgte die Phase, in der die OWL-S Ontologien mit der Domänenontologien angereichert werden sollten. Nach weiteren Fehlschlägen bei der Benutzung von Protege und CODE Entwicklungsumgebung wurde eine manuelle, leider sehr fehleranfällige Bearbeitung der OWL-S Dateien durchgeführt. Wie zur Beginn genannt, wurde die OWL-S VM gleichzeitig zur Validierung der OWL-S Dateien benutzt. Leider erwies sich äußerst schwierig diese ohne schwerwiegende Fehlermeldungen zu starten.

Zusammenfassend lässt sich sagen, dass die zur Beginn des Projektes spezifizierten Ziele betreffend der Realisierung nicht erreicht wurden. Im Einzelnen wurde kein vollständiger, funktionierender semantisch beschriebener Dienst implementiert. Außerdem wurde die für die Suche der SWS erforderliche Infrastruktur nicht realisiert.

5 Fazit

5.1 Zusammenfassung

Außer den Realisierungsvorgaben, standen in Rahmen des Ferienklubprojektes weitere Ziele im Vordergrund, welche zum Teil erreicht wurden. Einer dieser Ziele war die Sichtung der zurzeit vorhandenen Konzepte sowie die bereits entwickelten Hilfswerkzeuge. Hierfür war die Auseinandersetzung mit den zugrunde liegenden Grundlagen notwendig.

Mit diesem Wissen wurde ein Entwurf einer solchen Architektur durchgeführt. Hierzu würden die in der Forschung vorhandenen Ansätze untersucht und als Grundlage genommen. Dabei stand der Einsatz der WS und OWL-S Technik im Vordergrund.

Nachdem die theoretische Grundlage erarbeitet wurde und ein grober Entwurf stand, wurde eine Evaluierung der verfügbaren Werkzeuge durchgeführt. Hierbei wurden deren Reifegrad, Verfügbarkeit an Dokumentationen und die Akzeptanz durch die Community berücksichtigt. Zum anderen wurde darauf geachtet, dass alle in der Architektur vorgesehenen Komponenten frei verfügbar sein sollten.

Als Nächstes stand die Implementierungsphase auf dem Plan. Diese ist leider sehr kurz ausgefallen. Die hierbei entstandenen Schwierigkeiten werden im folgenden Abschnitt zusammengetragen und diskutiert.

5.2 Schlussfolgerungen und Kritik

Wie bereits erwähnt wurde in dieser Arbeit ein grober Entwurf der Architektur durchgeführt. Leider konnte eine notwendige Verfeinerung dieses nicht durchgeführt werden. Einer der Hauptgründe hierfür war die problembehaftete Benutzung der dort involvierten in Abschnitt 2 genannten, Komponenten und Werkzeuge.

Zur Beginn wurde Protege als eine Entwicklungsumgebung identifiziert, welche für alle bei der Entwicklung von SWS notwendigen Aufgaben eingesetzt werden sollte. Protege

eignete sich voll zur Entwicklung von Domänenontologien. Die ersten Probleme traten auf, als es um die Entwicklung der SWS ging. Hierfür notwendige Dokumentation war zum einen inkonsistent zu den gelieferten Beispielen. Zum anderen gab es keine Beschreibung der nicht intuitiven Oberfläche. Nach einigen Fehlversuchen wurde die Entwicklung der SWS mit Protege eingestellt.

Als Alternative stand die OWL-S IDE zur Verfügung. Diese biete die Möglichkeit SWS angefangen bei einfachen Java Anwendungen zu entwickeln. Dabei sollte diese dem Entwickler bis zum vollwertigen SWS unterstützen. Leider hatte sich während der Entwicklung herausgestellt, dass die Dokumentation ebenfalls inkonsistent zu den Leistungsfähigkeiten des Werkzeugs war. Dieses ging so weit, dass nicht nur die Beispiele schwierig zu starten waren, sondern dass die eigentliche Entwicklungsumgebung nicht die in der Beschreibung beschriebene Funktionalität bereitstellte.

Um möglichst viel bereits durchgeführter Arbeit wieder verwenden zu können, wurde als Nächstes auf einfache Werkzeuge zurückgegriffen. Dieser Weg ermöglichte die Erstellung von rohen OWL-S Beschreibungen. Aus dem in vorangegangenen Schritten gewonnen Erfahrungen motiviert, wurde zu manueller Vervollständigung (semantische Anreicherung) der OWL-S Beschreibungen übergegangen.

Im Abschnitt 2.4 beschriebene Laufzeitumgebung für OWL-S wurde dann anschließend dazu genutzt, die OWL-S Dateien zu validieren und die vorhandenen Informationen auszuwerten. Leider zeigte sich auch dort, dass die in (semwebcentral.org, 2005a) versprochene Funktionalität noch nicht einwandfrei funktionierte. Das Fehlen von jeglicher Dokumentation machte die fehlerlose Ausführung dieser Umgebung zunichte.

Bewertet man diese Arbeit aus Sicht von neu gewonnen Erkenntnissen, so wurde diese wie folgt ausfallen:

- Kritische Betrachtung: Bei einer wiederholten Durchführung dieser Arbeit wurde man die existierenden Ansätze zur Beginn auf die vorhandene Dokumentation und die hierbei laufenden Beispiele untersuchen. Es zeigte sich, dass die schnelle Entwicklung von neuen Ansätzen dazu neigt, keine bzw. mangelhafte Dokumentation zu liefern. Zum anderen wurden diese Ansätze so entwickelt, dass diese für bestimmte Beispiele (wenn vorhanden) spezialisiert wurden.

- Entwicklungsprozess: Da die Entwicklung der SWS einige Abstraktionsstufen durchläuft, ist eine Entwicklungsumgebung, welche den Entwickler durchweg bis zu fertige Implementierung unterstützt, unabdingbar. Dieses hat zum einem damit zutun, dass ein Wechsel der Umgebung Inkompatibilitätsprobleme bei den noch nicht finalisierten Formaten verursachen kann. Zum anderen versteht und konsequenterweise unterstützt jede Entwicklungsumgebung den Entwicklungsvorgang anders. Jeder Wechsel erschwert den Entwicklungsprozess.

5.3 Ausblick

Die in dieser Arbeit erreichten Ziele waren eher theoretischer und analytischer Art, sodass diese als eine Art Machbarkeitsstudie betrachtet werden kann. Im weiteren Verlauf des Projektes können die in dieser Arbeit gewonnen Erkenntnisse dazu dienen, die Entwicklung vergleichbarer semantischer Dienste und damit verbundene Risiken besser einschätzen zu können.

Zum einen würde durch diese Arbeit ein Katalog von zurzeit laufenden Entwicklungen gegeben. So stellt dieser einen Einstieg in die Welt der semantisch beschriebenen Dienste dar, was die zukünftige Rechergearbeit deutlich verkürzen kann.

Zum anderen stellt diese Grundüberlegungen zum möglichen Entwurf einer Suchplattform für SWS an. Die hier entworfene Architektur konnte nicht im Einzelnen evaluiert werden, diese stellt aber einen Einstieg dar, wie so ein System aufgebaut werden kann und welche Komponenten notwendig sind. Zum anderen wurden ansatzweise die Zuständigkeiten der einzelnen Komponenten und deren Beziehungen zu einander erarbeitet.

Aus Sicht des Ferienklubprojektes müsste als Erstes eine durchgängige Entwicklungsumgebung geschaffen werden. Dabei sollte eine klare Teilung der Aufgaben durchgeführt werden. In diesem Fall würde sich diese nach den jeweiligen Modulen richten. Dieses ist deshalb notwendig, da die Komponenten einen noch nicht hohen Reifegrad erreicht haben. Diese strikte Trennung und deren Einhaltung wurde die erfolgreiche Realisierung sehr begünstigen.

Als weitere Aufgabe ist die weitere Betrachtung und Analyse der verfügbaren Technologien zu sehen. Dieses wird aus verschiedenen Gründen motiviert. Zum einem sind während der Projektdurchführung Forschungsarbeiten wie (Balzer u. a., 2004) entstanden, die auf konzeptionelle Schwächen des OWL-S Ansatzes hinweisen. Diese Tatsache sollte nicht ungeachtet bleiben und möglicherweise zur Neubewertung der Eignung des Ansatzes führen.

Wie bereits erwähnt, leiden die in dieser Arbeit untersuchten Werkzeuge unter niedrigen Reifegrad. Bei Neubetrachtung dieser Problemstellung sollten möglicherweise weitere, ggf. mehr ausgereifte Entwicklungswerkzeuge identifiziert werden. So wird in (Gomez-Perez u. a., 2004) ein Überblick über eine Entwicklungsumgebung gegeben, welche ähnliche Funktionalität, wie die oben genannten Werkzeuge, anbieten soll. Aus zeitlichen Gründen konnte diese Möglichkeit leider nicht weiterverfolgt werden.

Zur Beginn dieser Ausarbeitung wurde darauf hingewiesen, dass für die Suche relevante Merkmale eines Dienstes in der OWL-S Ontologie hauptsächlich durch vier Merkmale definiert werden können. Gemeint sind an diese Stelle die Eingaben, Ausgaben, Vorbedingung und Auswirkungen eines Dienstes. In den Arbeiten (Zhou u. a., 2005) und (Zhou u. a., 2004) würde dies als Schwäche von OWL-S identifiziert. Als Konsequenz hierfür wurde in den gleichnamigen Arbeiten eine Ontologie zur Beschreibung von Quality of Service Merkmalen entwickelt, sowie deren Bewertungskriterien. Auch diese Möglichkeit sollte nicht unberücksichtigt bleiben, wenn es um die Verbesserung der Dienstsuche geht.

Literatur

[Apache.org 2005] APACHE.ORG: *jUDDI*. 2005. – URL `http://ws.apache.org/juddi/`. – OWL-S VM (15.02.2006)

[Avar u. a. 2005] AVAR, Andras ; CHAPPELL, David ; DANIELS, Glen u. a.: *Apache Axis Version 1.3*. 2005. – URL `http://ws.apache.org/axis/`. – Open Source Web Services Container (21.12.2005)

[Balzer u. a. 2004] BALZER, Steffen ; LIEBIG, Thorsten ; WAGNER, Matthias: Pitfalls of OWL-S: a practical semantic web use case. In: *ICSOC*, 2004, S. 289–298

[Gomez-Perez u. a. 2004] GOMEZ-PEREZ, Gomez-Perez ; GONZáLEZ-CABERO, Rafael ; LAMA, Manuel: ODE SWS: A Framework for Designing and Composing Semantic Web Services. In: *IEEE Intelligent Systems* 19 (2004), Nr. 4, S. 24–31. – ISSN 1094-7167

[graphviz.org 2005] GRAPHVIZ.ORG: *Graphviz - Graph Visualization Software*. 2005. – URL `http://www.graphviz.org/`. – Graphviz - Graph Visualization Software für Protege (15.02.2006)

[OWL-S 2005] OWL-S: *OWL-based Web Service Ontology*. 2005. – URL `http://www.daml.org/services/owl-s/`

[Paolucci u. a. 2003] PAOLUCCI, Massimo ; ANKOLEKAR, Anupriya ; SRINIVASAN, Naveen ; SYCARA, Katia P.: The DAML-S Virtual Machine. In: *International Semantic Web Conference*, 2003, S. 290–305

[Paolucci u. a. 2002a] PAOLUCCI, Massimo ; KAWAMURA, Takahiro ; PAYNE, Terry R. ; SYCARA, Katia P.: Importing the Semantic Web in UDDI. In: *CAiSE '02/ WES '02: Revised Papers from the International Workshop on Web Services, E-Business, and the Semantic Web*. London, UK : Springer-Verlag, 2002, S. 225–236. – ISBN 3-540-00198-0

[Paolucci u. a. 2002b] PAOLUCCI, Massimo ; KAWAMURA, Takahiro ; PAYNE, Terry R. ; SYCARA, Katia P.: Semantic Matching of Web Services Capabilities. In: *International Semantic Web Conference*, 2002, S. 333–347

[RACER-Systems 2005] RACER-SYSTEMS: *RACER*. 2005. – URL `http://www.racer-systems.com/`. – RACER Inferenzemaschine (15.02.2006)

[semwebcentral.org 2005a] SEMWEBCENTRAL.ORG: *Java 2 OWL-S Converter*. 2005. – URL `http://java2owl-s.projects.semwebcentral.org/`. – Java 2 OWL-S Converter (15.02.2006)

[semwebcentral.org 2005b] SEMWEBCENTRAL.ORG: *OWL-S 2 UDDI Converter*. 2005. – URL http://owl-s2uddi.projects.semwebcentral.org/. – OWL-S 2 UDDI Converter (15.02.2006)

[Semwebcentral.org 2005] SEMWEBCENTRAL.ORG: *The OWL-S Editor*. 2005. – URL http://owlseditor.semwebcentral.org/. – The OWL-S Editor für Protege (15.02.2006)

[semwebcentral.org 2005a] SEMWEBCENTRAL.ORG: *OWL-S VM*. 2005. – URL http://projects.semwebcentral.org/projects/owl-s-vm/. – OWL-S VM (15.02.2006)

[semwebcentral.org 2005b] SEMWEBCENTRAL.ORG: *WSDL 2 OWL-S*. 2005. – URL http://www.daml.ri.cmu.edu/wsdl2owls/. – WSDL 2 OWL-S (15.02.2006)

[Srinivasan u. a. 2005] SRINIVASAN, Naveen ; PAOLUCCI, Massimo ; SYCARA, Katia: CODE: A Development Environment for OWL-S Web services / Robotics Institute, Carnegie Mellon University. Pittsburgh, PA, October 2005 (CMU-RI-TR-05-48). – Forschungsbericht

[Srinivasan u. a. 2006] SRINIVASAN, Naveen ; PAOLUCCI, Massimo ; SYCARA, Katia: Semantic Web Service Discovery in the OWL-S IDE. In: *HICSS '06: Proceedings of the Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06) Track 6*. Washington, DC, USA : IEEE Computer Society, 2006, S. 109.2. – ISBN 0-7695-2507-5

[Srinivasan u. a. 2004] SRINIVASAN, Naveen ; PAOLUCCI, Massimo ; SYCARA, Katia P.: An Efficient Algorithm for OWL-S Based Semantic Search in UDDI. In: *SWSWPC*, 2004, S. 96–110

[Stanford 2005] STANFORD, University: *Protege Ontology Editor and Knowledge Acquisition System*. 2005. – URL http://protege.stanford.edu/. – Protege Ontology Editor and Knowledge Acquisition System (15.02.2006)

[W3C 2004] W3C (Hrsg.): *OWL Web Ontology Language*. 2004. – URL http://www.w3.org/TR/owl-features/. – Zugriffsdatum: 10-04-2005

[Wendt 2005a] WENDT, Piotr: *Semantic Web Service Discovery*. Dezember 2005. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master05-06/wendt/abstract.pdf. – Ausarbeitung zum Seminar (SR) des Master-Studiengangs Informatik an der HAW Hamburg (15.02.2006)

[Wendt 2005b] WENDT, Piotr: *Semantic Web Services*. Juli 2005. – URL http://users.informatik.haw-hamburg.de/. – Ausarbeitung zum Seminar Anwendungen 1 des Master-Studiengangs Informatik an der HAW Hamburg (15.02.2006)

[Zhou u. a. 2004] ZHOU, Chen ; CHIA, Liang-Tien ; LEE, Bu-Sung: DAML-QoS Ontology for Web Services. In: *ICWS '04: Proceedings of the IEEE International Conference on Web Services (ICWS'04)*. Washington, DC, USA : IEEE Computer Society, 2004, S. 472. – ISBN 0-7695-2167-3

[Zhou u. a. 2005] ZHOU, Chen ; CHIA, Liang-Tien ; LEE, Bu-Sung: Service discovery and measurement based on DAML-QoS ontology. In: *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*. New York, NY, USA : ACM Press, 2005, S. 1070–1071. – ISBN 1-59593-051-5