Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Projektbericht

Gerrit Diederichs

Aufbau einer Semantic Web Infrastruktur und Integration von externen Informationsdiensten

Betreuender Prüfer: Prof. Dr. Kai von Luck

Fakultät Informatik
Studiendepartment Computer Science

Gerrit Diederichs

Thema des Projektbericht

Aufbau einer Semantic Web Infrastruktur und Integration von externen Informationsdiensten

Stichworte

Semantic Web, RDF, Ontologien, OWL, Inferenzmaschinen

Kurzzusammenfassung

In diesem Artikel handelt es sich um einen Bericht über ein Projekt, an dem der Autor im Rahmen des dritten Semesters des Masterstudienganges teilgenommen hat. Das Projekt hat sich dabei im Rahmen eines fiktiven Ferienclub-Szenarios bewegt, für das verschiedenste Anwendungen implementiert werden konnten. Der Autor hat sich in diesem Rahmen mit der Semantic Web Technologie beschäftigt, um ein Informationsportal für Clubbesucher aufzubauen. Hierbei sollten auch externe Dienstleister mittels Semantic Web Technologien eingebunden werden. Der Autor hat auf Basis eines geeigneten Szenarios eine auf Ontologien basierende Architektur entworfen und prototypisch realisiert. Im Folgenden werden die verfolgten Ziele, sowie der Entwurf und die Implementierung einer prototypischen Webanwendung beschrieben. Die im Laufe des Projektes gesammelten Erfahrungen, sowie die notwendigen weiteren Schritte, um den Prototypen auszubauen werden im obligatorischen Fazit diskutiert.

Inhaltsverzeichnis

1 Einleitung

In der Veranstaltung „Projekt Angewandte Informatik“ des dritten Semesters im Masterstudiengang an der HAW Hamburg im Wintersemester 2005/2006 hatten die Studenten die Möglichkeit im Rahmen eines fiktiven Ferienclub-Szenarios verschiedene Technologien zu erproben. Der Autor hat sich hier mit der Technologie des Semantic Web auseinandergesetzt. Ziel des Semantic Web ist es Internetressourcen mit einer formalen Semantik zu beschreiben und sie so durch Maschinen besser verarbeitbar zu machen. Basis hierfür sind einerseits Ontologien - formale Wissensrepräsentationen einer bestimmten Wissensdomäne - und andererseits das Resource Description Framework (RDF), um die Ressourcen zu annotieren.In vorangegangenen Veranstaltungen des Masterstudienganges hatte der Autor bereits Gelegenheit sich in theoretische Aspekte des Semantic Web einzuarbeiten (Diederichs, 2005a) und (Diederichs, 2005b) im Rahmen des Projektes sollten nun praktische Erfahrungen in diesem Bereich gesammelt werden. Es wurden folgende Ziele verfolgt:

- Evaluierung verschiedener Werkzeuge für die Entwicklung von Semantic Web Komponenten
- Praktische Einarbeitung in Semantic Web Technologie
- Vorbereitung auf eine Masterthesis im Bereich Semantic Web

Das Projekt war lehrplanmäßig auf einen Umfang von ca. 100 Stunden angelegt. Aufgrund dieser begrenzten Zeit fiel die Entscheidung zunächst nur prototypisch eine Semantic Web Anwendung zu realisieren, anstatt wie ursprünglich geplant eine voll funktionsfähige Webanwendung in Form eines Informationsportals zu realisieren. Im Rahmen dieses Projektberichts soll zunächst ein kurzer Überblick über die Basistechnologien des Semantic Web gegeben werden (Kapitel 2). In Kapitel 3 wird das gewählte Szenario sowie die gesetzten Meilensteine für das Projekt vorgestellt. Für die Planung und Realisierung des Prototypen wurden bereits vorhandene Komponenten verwendet. Diese werden in Kapitel 4 vorgestellt. Der Entwurf (Kapitel 5)und die relevanten Details der prototypischen Realisierung (Kapitel 6)werden beschrieben. Abschließend werden in Kapitel 7 die gesammelten Erfahrungen, sowie der erreichte Stand des Prototypen dargelegt und durch einen Ausblick auf die nächsten Schritte zur Weiterentwicklung abgerundet.

2 Basistechnologien

In diesem Abschnitt werden das Resource Description Framework (RDF) und die Ontologiesprache Web Ontology Language (OWL) kurz vorgestellt. Mithilfe dieser beiden Technologien lassen sich Wissensbasen erstellen, die die Grundlage für das Semantic Web bilden. Eine Wissensbasis im Kontext des Semantic Web besteht aus dem Wissensmodell das die entsprechende Wissensdomäne logisch beschreibt (die Ontologie) und Individuen die bestimmten Konzepten des Wissensmodells aufgrund ihrer Eigenschaften zugeordnet werden können. Ausführlichere Quellen zu diesem Thema lassen sich unter anderem hier finden (Michael C. Daconta, 2003), (Horrocks)

2.1 Resource Description Framework RDF

Das Resource Description Framework (RDF)ist eine W3C Spezifikation (RDFW3C).Es beschreibt ein Metadatenmodell für beliebige Ressourcen. Das Modell besteht dabei aus Aussagen über Ressourcen. Eine RDF-Aussage ist aufgebaut als ein Triple mit Subjekt, Prädikat und Objekt. Dabei kann jeder Bestandteil des Triples eine Ressource sein. Eine RDF-Aussage lässt sich als gerichteter Graph interpretieren. Das Objekt einer Aussage kann wiederum eine Aussage sein. Man spricht hierbei von Reification. RDF-Graphen werden mittels einer XML-basierten Notation serialisiert. Mit RDF können Ressourcen und Beziehungen zwischen Ressourcen beschrieben werden. Im Semantic Web wird RDF verwendet um die Individuen einer Wissensbasis zu beschreiben.

2.2 Ontologiesprachen und OWL

Ontologien dienen zur formalen Repräsentation von Wissen. Eine häufig zitierte Definition des Begriffes Ontologie im informationstechnischen Kontext ist die von Gruber:

> An ontology is a formal, explicit specification of a shared conceptualization. (Gruber, 1993)

Ontologiesprachen sind die Werkzeuge um Ontologien zu modellieren, also die Wissensdomäne für einen bestimmten Kontext zu formalisieren und logisch zu beschreiben. Das Paradigma der Ontologiemodellierung ähnelt dem der objektorientierten Softwaremodellierung. Eine Ontologie besteht aus verschiedenen Konzepten (auch Klassen genannt), die man mit Klassen in der Objektorientierung vergleichen kann, und ihren Beziehungen untereinander. Konzepte haben Eigenschaften die das Konzept charakterisieren. Eine Eigenschaft kann dabei durch einen primitiven Datentyp (Datatype Property) beschrieben werden oder durch eine Beziehung zu einem anderen Konzept (Object Property). Neben dem Erstellen eigener Ontologien können Ontologiesprachen auch dazu genutzt werden unterschiedliche Ontologien aufeinander zu mappen. Dies geschieht über Äquivalenzaussagen bezüglich der Konzepte der zu mappenden Ontologien. Die Ontologiesprache Web Ontology Language (OWLW3C) ist eine Recommendation des W3C für das Semantic Web und stellt somit einen

quasi Standard dar. OWL ist aus DAML+OIL hervorgegangen. OWL baut auf RDF beziehungsweise RDF-Schema auf. Ein Individuum eines Konzeptes in OWL wird beispielsweise über <rdf:type> erzeugt. OWL verfügt über drei unterschiedliche Ausdrucksstärken bezüglich der Modellierung von Ontologien. OWL Lite < OWL Description Logic < OWL Full. Hierbei gilt, dass die Mächtigkeit der Sprachkonstrukte von OWL Lite bis hin zu OWL Full ansteigt und der Sprachumfang der jeweils mächtigeren Sprache die der darunterliegenden voll enthält. Je mächtiger die verwendeten Sprachkonstrukte sind, desto aufwändiger ist ein späteres Reasoning (Inferenz) über die Ontologie. Hierbei gilt, dass OWL Description Logic noch entscheidbar ist, OWL Full jedoch Konstrukte enthalten kann die nicht mehr entscheidbar sind. Generell muss man bei der Modellierung von Ontologien stets einen Kompromiss zwischen Ausdrucksstärke und Entscheidbarkeit der Ontologie finden. Meist werden Ontologien höchstens mit den Konstrukten der Beschreibungslogik (Description Logic) - einem entscheidbaren Fragment der Prädikatenlogik - modelliert, um einen vernünftigen Kompromiss zu erzielen.

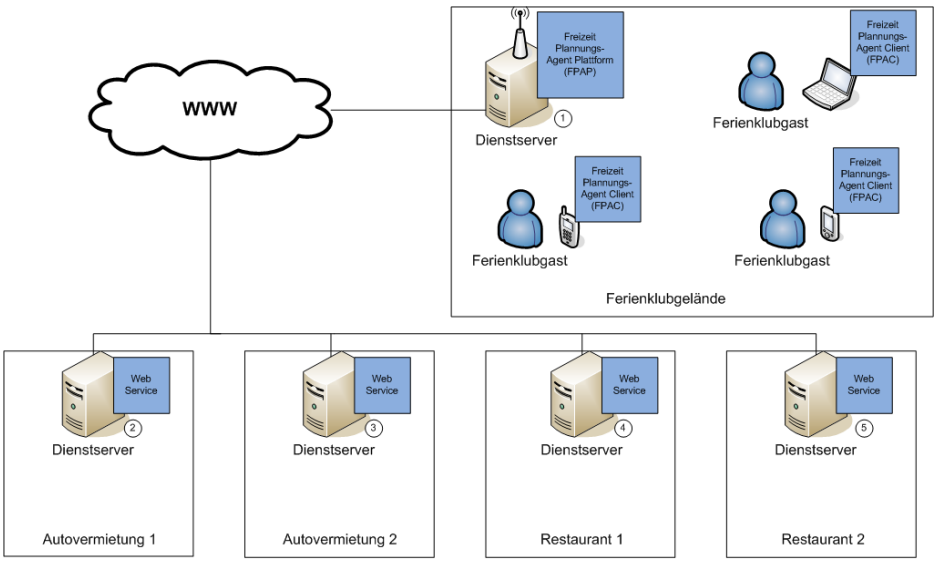

Abbildung 1: Szenario Ferienclub

3 Projektszenario

Für den Einsatz von Semantic Web Technologien hat die Semantic Web Gruppe des Projektes bestehend aus Piotr Wendt, Artem Khvat, Thomas Steinberg und dem Autor das in Abbildung 1 dargestellte Szenario erarbeitet. Kern ist die Freizeit Planungs Agent Plattform (FPAP) getaufte Komponente. Der Name ist ein wenig überfrachtet, da es sich hierbei zunächst nur um eine Webanwendung handelt, die einerseits eigene Dienste, d.h. Dienste des Ferienclub, und externe Dienste für Besucher des Ferienclub auf Anfrage liefert. In dem Szenario bestehen die Dienste aus Autovermietungen und Restaurants. Die Dienste basieren dabei auf einer Semantic Web Infrastruktur, die Angebote werden also in Wissensbasen gespeichert. Die FPAP-Komponente bietet einerseits eine Suche innerhalb der eigenen Wissensbasis an und andererseits ein Gateway für die Suche in den Wissensbasen der externen Dienstleister. Die Dienste sollen dabei als Web Services zur Verfügung stehen. Ferienclubbesucher können über eine webbasierte Benutzerschnittstelle nach passenden Angeboten suchen.

3.1 Meilensteine

Das Projekt lief über einen Zeitraum von 16 Wochen mit 8 Semesterwochenstunden (d.h. 6 Zeitstunden). Zur Risikoabschätzung wurden vier Meilensteine gesetzt um den Fortschritt des Projekts zu bewerten. Ein Meilenstein bewegte sich im Zeitraum von 4 Wochen. Für den eigenen Beitrag zum Projekt hat der Autor folgende 4 Meilensteine gesetzt:
Meilenstein 1 enthielt folgende Punkte:

- Entwicklung des Ferienclub-Szenarios

- Auswahl geeigneter Werkzeuge, um Wissensbasen zu erstellen und Inferenz auf diesen Wissenbasen durchzuführen.
- Einarbeitung in die APIs für den Zugriff auf die Wissensbasen und Inferenz (als Ergebnis ist die Datei OWLAPIDemoApplication.java enstanden).
- Erstellung einer einfachen Ferienclub-Ontologie (Taxonomie) für Mietwagen die auch als Referenz für das Mapping auf die externen Ontologien dient.
- Entwurf der Basisarchitektur für den Prototypen

Meilenstein 2 enthielt folgende Punkte:

- Erstellung einer Ontologie eines externen Mietautoanbieters mit gleicher Struktur wie die Referenz.
- Webbasierter Prototyp der zunächst nur die Daten der Ferienclub-Ontologie durchsucht.

Meilenstein 3 enthielt folgende Punkte:

- Verfeinerung der Konzepte der Ontologien um Restriktionen sprich Regeln (Ein Motorrad hat höchstens 2 Sitzplätze, Ein Oberklasse PKW hat mindestens 250 PS etc.)
- Anbindung einer Inferenzmaschine
- Integration der externen Wissensbasis in den Prototypen

Meilenstein 4 enthielt folgende Punkte:

- Integration der Komponenten des Prototypen in die Gesamtarchitketur (SOA) des Projektes
- Integration einer Mapping Komponente für Ontologien, die im Rahmen des Projektes von Artem Khvat entwickelt wurde.

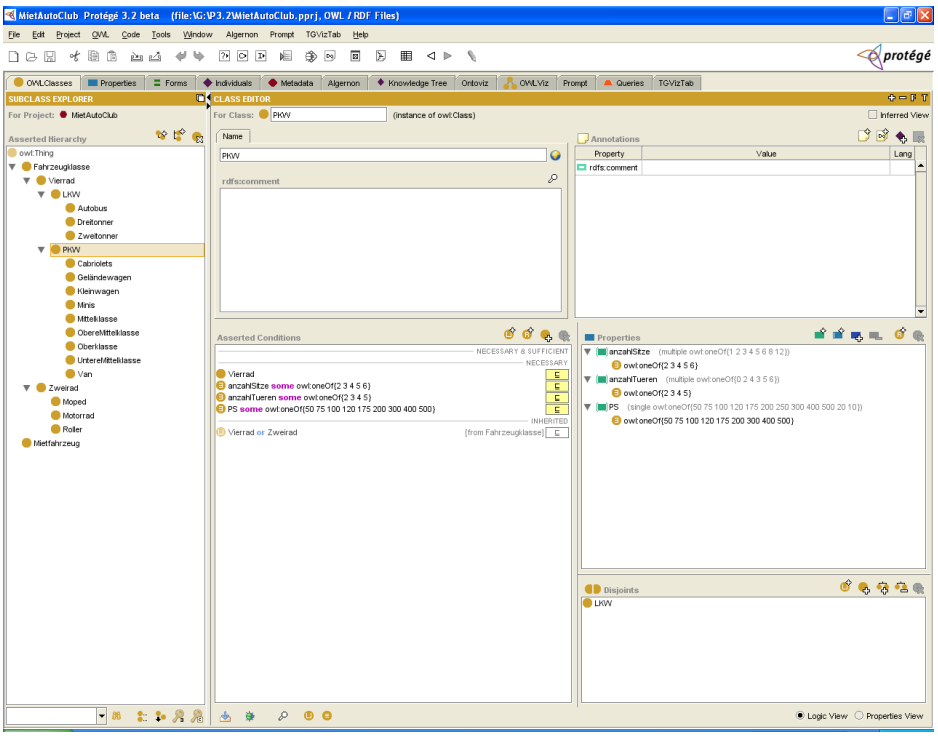

Abbildung 2: Protégé OWL

4 Verwendete Komponenten

Der Prototyp sollte ein Webanwendung sein, die auf einer semantischen, auf Ontologien basierenden Infrastruktur (Backend) aufsetzt. Für die Entwicklung der Ontologien (Ontology Engineering) wurde Protégé in der Version 3.2 beta (Protégé) eingesetzt.

Protégé ist ein Open Source Projekt der Stanford Universität. Es ist in Java geschrieben und stellt eine Entwicklungsumgebung für Ontologien bereit. Protégé ist über Plugins erweiterbar und mit der Entwicklungsumgebung Eclipse vergleichbar. Das ursprüngliche Protégé benutzt eine auf Frame Logic basierende Ontologiesprache. Im Projekt sollte allerdings die Web Ontology Language (OWL) eingesetzt werden, da OWL, wie bereits in Kapitel 2.2 erwähnt, einen quasi Standard darstellt. Für die Entwicklung von Ontologien in OWL bietet Protégé das Protégé-OWL-Plugin, das im Projekt verwendet wurde. Für das spätere Mapping von Ontologien sollte das AnchorPrompt-Plugin eingesetzt werden.

Um das Potential einer Ontologie voll auszunutzen werden Inferenzmaschinen verwendet. Sie leiten neues Wissen aus einer bestehenden Ontologie ab (Deduktion). Darüberhinaus sichern sie die Konsistenz einer Ontologie, d.h. ihrer Konzepte, und dienen zur Klassifikation von unbekannten Individuen oder Konzepten. Schließlich lassen sich über Inferenzmaschinen auch Äquivalenzen zwischen Konzepten ermitteln. Im Rahmen des Projekts wurde die Inferenzmaschine RACERPro (RACER) eingesetzt. RACERPro ist eine Inferenzmaschine die auf Basis der Beschreibungslogik arbeitet. RACER steht für **R**enamed **A**Box and **C**oncept **E**xpression **R**easoner. Ursprünglich wurde die Inferenzmaschine unter dem Na-

men RACER als Forschungsprojekt an der Technischen Universität Hamburg-Harburg entwickelt.Inzwischen wird die Inferenzmaschine unter dem Namen RacerPro von der Racer Systems GmbH und Co. KG als kommerzielles Produkt angeboten. Sie ist aber im Rahmen einer zeitlich begrenzten Educational License frei verfügbar.
Die prototypische Webanwendung, die auf der semantischen Infrastruktur aufsetzt wurde als JSP/Servlet-Anwendung mit dem Webframework Struts (Apache, a) entwickelt. Als Webcontainer wurde Apache Tomcat 5.5 (Apache, b) eingesetzt. Der Prototyp wurde mit der Open Source IDE Eclipse (Eclipse) in Java entwickelt.

5 Entwurf

In Abbildung 3 ist die grobe Architektur aufgeführt. Die semantische Schicht (Semantic Layer) enthält die Wissensbasis bestehend aus Ontologien und deren Individuen, sowie einer Inferenzmaschine. Eine Besonderheit bei der Entwicklung einer Semantic Web Anwendung ist das Ontologie Engineering. Es umfasst Aufgaben wie die eigentliche Entwicklung der Ontologie, aber auch das Mapping zwischen Ontologien und die Überprüfung der Konsistenz einer Ontologie. Im Projekt wurde dafür das in Kapitel 4 vorgestellte Werkzeug Protégé eingesetzt. Im Application Layer wird auf diese Infrastruktur zugegriffen. Für den Zugriff auf die Wissensbasen wurde die Protégé-OWL-API genutzt. Diese baut auf der Jena 2 API (Jena2) auf. In der Control Logic werden die Wissensbasen entsprechend der Benutzeranfragen durchsucht und die Ergebnisse zurückgeliefert. Für den Zugriff auf die externen Wissensbasen gibt es eine Mediator Komponente, die als Gateway zu den externen Wissensbasen fungiert. Sie verwendet Mapping Regeln, um die Anfragen für die externen Wissensbasen aufzubereiten. Die Mapping Regeln werden dabei vom Ontologie Engineer mit einem geeignetem Werkzeug halbautomatisch erzeugt. Ein Beispiel für so ein Werkzeug ist das Prompt-Plugin für Protégé. Das Prompt-Plugin wird verwendet um Ontologien zu verschmelzen (Ontology Merging) oder um Ontologien zu mappen (Ontology Mapping). Für das Mapping enthält das Prompt-Plugin ein Feature namens AnchorPrompt. AnchorPrompt vergleicht die Ontologien dabei auf struktureller Ebene. Eine Ontologie lässt sich von der Topologie her als Baumstruktur darstellen. Ein Ontologie Engineer kann mittels AnchorPrompt einige „Anker“ zwischen den beiden Ontologien setzen. Das heißt er mappt einige Knoten der beiden Ontologien manuell. Anhand dieser initialen, manuell vorgenommenen Mappings versucht AnchorPrompt die restlichen Konzepte (die Knoten des Baumes) automatisch zu mappen. Abbildung 4 zeigt die prinzipielle Funktionsweise von AnchorPrompt.

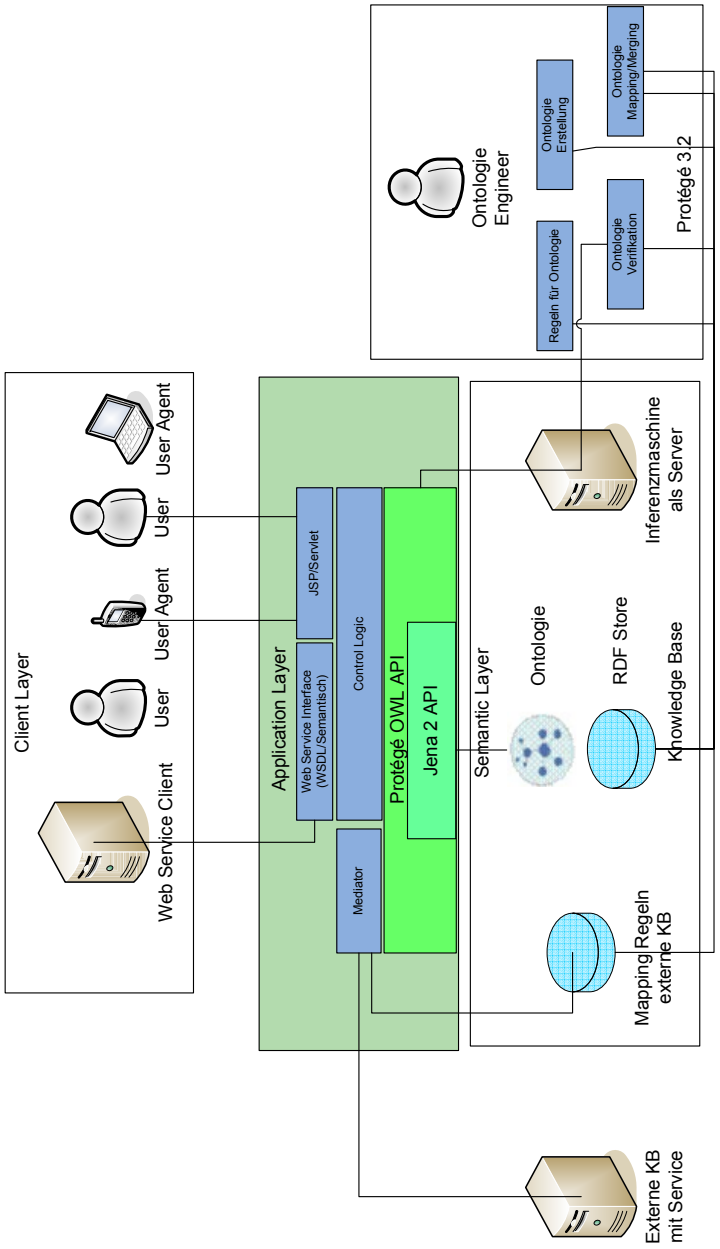

Abbildung 3: Architektur Projekt

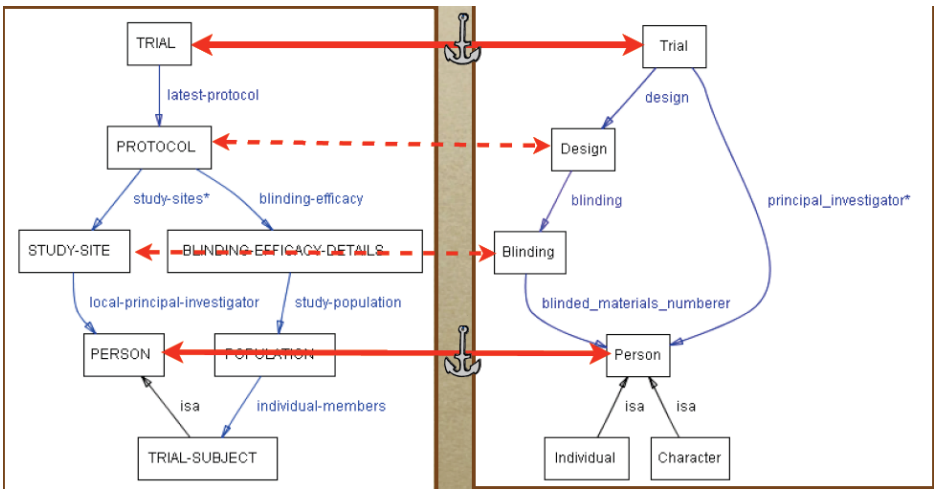

Abbildung 4: Prinzip von AnchorPrompt

6 Realisierung des Prototypen

Für die Realisierung des Prototypen wurden zunächst die Ontologien entwickelt, auf deren Basis der Prototyp nach Angeboten suchen sollte. Anschließend wurde der eigentliche Prototyp realisiert. Dies geschah in zwei Stufen zunächst nur für die Suche in der Wissensbasis des Ferienclub. Später auch in der Wissensbasis eines externen Anbieters.

6.1 Entwicklung der Ontologien

Genau wie Software werden auch Ontologien iterativ entwickelt. Zunächst sollten die Ontologien nur einfache Taxonomien - eine Klassifikation der Konzepte innerhalb einer Vererbungshierarchie - sein, ohne spezifische Eigenschaften der Konzepte. Dabei wurden zwei Ontologien für Autovermietungen entwickelt, die sich strukturell gleichen allerdings mit unterschiedlichen Benennungen der Konzepte ausgestattet sind. Die Referenzontologie war hierbei die Ontologie des Ferienclub. Im nächsten Schritt sollte eine dritte Ontologie für Autovermietungen mit einer anderen Topologie entwickelt werden. Abbildung 5 sind die beiden Ontologien dargestellt. Im nächsten Schritt wurden die Ontologien um Restriktionen bzw. Regeln erweitert. Dafür wurden die Konzepte sowohl um Eigenschaften erweitert, als auch um Restriktionen, welche Werte diese Eigenschaften haben dürfen. Folgender OWL Ausschnitt beschreibt beispielsweise die Regel „Ein Konzept mit dem Namen Van, hat eine Eigenschaft PS die einen Wert aus der Menge (100 120 175 200) annehmen kann“:

```
<owl:Class rdf:ID="Van">
<rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:about="#PS"/>
        </owl:onProperty>
```

```
        <owl:someValuesFrom>
          <owl:DataRange>
            <owl:oneOf rdf:parseType="Resource">
              <rdf:rest rdf:parseType="Resource">
                <rdf:first rdf:datatype=
                "http://www.w3.org/2001/XMLSchema#int"
                >120</rdf:first>
                <rdf:rest rdf:parseType="Resource">
                  <rdf:rest rdf:parseType="Resource">
                    <rdf:rest rdf:resource=
                    "http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                    <rdf:first rdf:datatype=
                    "http://www.w3.org/2001/XMLSchema#int"
                    >200</rdf:first>
                  </rdf:rest>
                  <rdf:first rdf:datatype=
                  "http://www.w3.org/2001/XMLSchema#int"
                  >175</rdf:first>
                </rdf:rest>
              </rdf:rest>
              <rdf:first rdf:datatype=
              "http://www.w3.org/2001/XMLSchema#int"
              >100</rdf:first>
            </owl:oneOf>
          </owl:DataRange>
        </owl:someValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
```

Auf Basis zunächst nur der Ferienclub Ontologie wurde dann mit der Entwicklung des Prototypen begonnen.

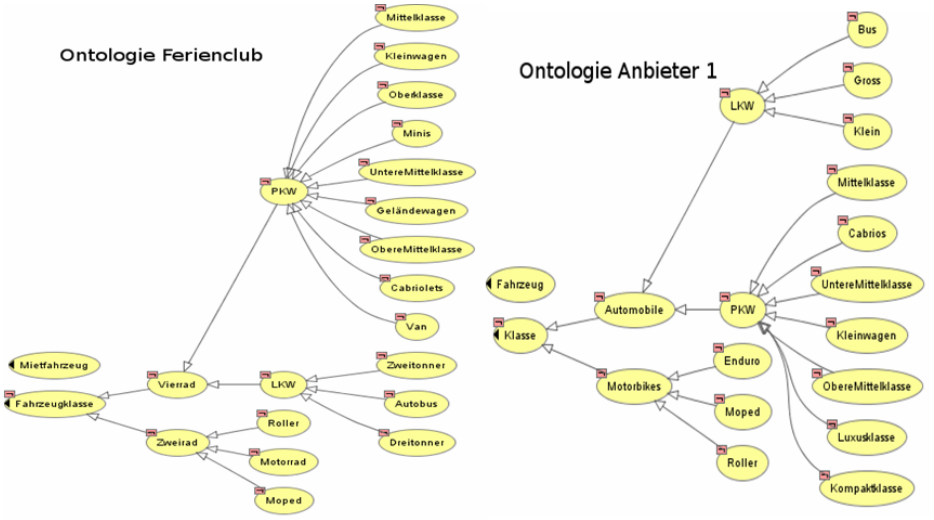

Abbildung 5: Ontologie Ferienclub und Anbieter 1

6.2 Entwicklung des Prototypen

Abbildung 6 zeigt die wesentlichen Komponenten des Prototypen der Freizeit Planungs Agenten Plattform (FPAP), die in Kapitel 3 vorgestellt wurde. Der Prototyp sollte in mehreren Stufen ausgebaut werden. In der ersten Stufe wurde zunächst die Wissensbasis des Ferienclubs anhand der Fahrzeugklassen durchsucht. Das JSP Frontend bietet hierfür eine Einstiegsmaske, in der Fahrzeugklassen von Interesse ausgewählt werden können. Die Komponente SearchService bietet eine Methode `getCarOffersFromKnowledgebaseBaseByVehicleClass(vehicleClass, kb)`. Sie gibt alle Angebote zurück, die der gesuchten Fahrzeugklasse oder Unterklassen entsprechen. Die Klasse `BasicSearchServiceImpl` implementiert diese Suche. Hierfür wird zunächst festgestellt welche Fahrzeugklassen durch die gesuchte Fahrzeugklasse subsumiert werden.

```
OWLNamedClass baseVehicleClass =
protegeModel.getOWLNamedClass(vehicleClass);
Collection vehicleClasses =
baseVehicleClass.getNamedSubclasses(true);
vehicleClasses.add(baseVehicleClass);
```

Anschließend wird geprüft, welche Individuen diesen Fahrzeugklassen angehören.

```
protected boolean
 isIndividualInSearchSet(OWLIndividual individual, vehicleClasses) {
Collection props = individual.getRDFProperties();
for (Iterator jt = props.iterator(); jt.hasNext();) {
RDFProperty property = (RDFProperty) jt.next();
if(property instanceof OWLObjectProperty) {
```

```
OWLIndividual carClass =
(OWLIndividual)individual.getPropertyValue(property);
OWLNamedClass individualClass =
(OWLNamedClass)carClass.getProtegeType();
if (vehicleClasses.contains(individualClass))
return true;
}
}
return false;
}
```

Da die Ontologie als einzige ObjectProperty die Fahrzeugklasse hat (alle anderen Properties sind DatatypeProperties), ist klar, dass eine gefundene Instanz einer ObjectProperty die der Fahrzeugklasse sein muss. Individuen, die der Suchanfrage entsprechen werden als Ergebnis in Form einer Collection zurückgegeben. In der zweiten Stufe des Prototypen wurde die zweite erstellte Ontologie von „MietautoAnbieter 1“ angebunden. Dies geschah über die Mediator Komponente. Hier war ursprünglich geplant das Werkzeug AnchorPrompt zu verwenden, um das Mapping zwischen den Ontologien halbautomatisch zu erstellen. Leider hat sich herausgestellt, dass AnchorPrompt kein sinnvoll weiter zu verwendendes Output File für ein durchgeführtes Mapping generiert. Daher wurde für den Prototypen zunächst eine Properties Datei manuell angelegt, die das Mapping enthält. Mithilfe des Mappings bearbeitet der Mediator die Suchanfrage so, dass sie für die Ontologie des externen Autoanbieters verständlich ist. Weitere Stufen für den Ausbau des Prototypen waren geplant (siehe Kapitel 3.1) wurden jedoch aus Zeitgründen nicht mehr umgesetzt.

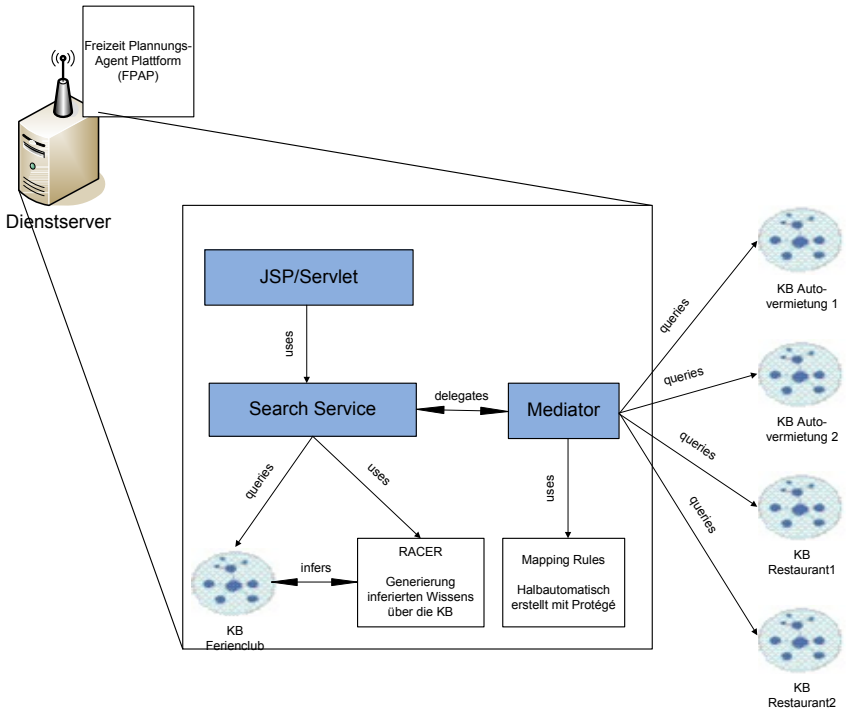

Abbildung 6: Realisierung der FPAP Komponente

7 Fazit

Das abschließende Fazit ist unterteilt in drei Abschnitte. Zunächst wird der derzeitige Stand der entwickelten Ontologien und des Prototypen dargelegt. Anschließend werden die gesammelten Erfahrungen aus dem Projekt beschrieben und schließlich ein Ausblick bezüglich der Weiterentwicklung des Prototypen und eine Einbettung in das Gesamtszenario gegeben.

7.1 Stand Ontologien und Prototyp

Im ursprünglichen Projektszenario war die Entwicklung von fünf Ontologien geplant. Eine Ferienclub Ontlogie als Referenz und jeweils zwei Ontologien für externe Autovermietungen beziehungsweise Restaurants. Dabei jeweils eine Ontologie die strukturell gleich der Referenzontologie ist und eine die sich in der Struktur unterscheidet. Momentan gibt es erst zwei Ontologien. Nachdem die Einarbeitung in ein Werkzeug wie Protégé aber abgeschlossen ist, ließen sich die übrigen Ontologien wahrscheinlich in relativ kurzer Zeit entwickeln. Leider hat der Prototyp nicht die Funktionalität erreicht, die ursprünglich gemäß der gesetzten Meilensteine geplant war. Der momentane Stand des Prototypen liegt zwischen Meilenstein 2 und 3. Der Prototyp implementiert derzeit die Funktionalität die zwei erstellten Ontologien zu durchsuchen. Hierfür kann die gewünschte Fahrzeugklasse als Suchkriterium angegeben werden. Bisher ist keine Unterstützung für die Nutzung einer externen Inferenzmaschine wie RacerPro implementiert. Für die Suche nach den Fahrzeugklassen wird die in der Protégé-OWL-API eingebaute Subsumption über die Instanzmethode `OWLNamedClass.getNamedSubclasses(true)` verwendet. Die Inferenzmaschine RacerPro wurde lediglich im Rahmen des Ontologie Engineerings eingesetzt, um die Konsistenz der Ontologien sicherzustellen. Die einzelnen Komponenten des Prototypen sind momentan nicht in die ursprünglich vom Architekturteam geplante Service Oriented Architecture (SOA) eingebunden, sondern liegen auf einem Web Container. Schließlich ist das bisherige Mapping manuell und nur provisorisch. Das hierfür in Protégé vorgesehene Werkzeug AnchorPrompt war nicht für den beabsichtigten Zweck geeignet. Artem Khvat hat allerdings im Rahmen seines Projektbeitrags ein Werkzeug entwickelt, das ebenfalls ein Mapping von Ontologien durchführt. Eventuell eignet es sich als Ersatz für Protégé's AnchorPrompt.

7.2 Lessons learned

Wie eingangs in Kapitel 1 erwähnt war eines der Ziele die Evaluierung bestehender Werkzeuge für das Semantic Web. Hierbei wurde vor allem mit dem Werkzeug Protégé gearbeitet. Das Fazit fällt gemischt aus. Das Protégé-OWL-Plugin ist zum Entwickeln von Ontologien recht gut geeignet und funktioniert im Wesentlichen erwartungsgemäß. Das Prompt-Plugin hingegen konnte bezüglich des Mappings von Ontologien nicht überzeugen. Brauchbare Tutorials bezüglich der Entwicklung von Ontologien mit Protégé sind in begrenztem Umfang vorhanden z.B. (Matthew Horridge, 2004). Trotzdem ist die Einarbeitung in diesem Bereich recht mühselig und hat auch relativ viel Zeit gekostet. Die verwendeten APIs jedoch sind schlecht bis garnicht dokumentiert. Es gibt lediglich einige wenige Code-Beispiele, die aber

nicht ausreichen, um einen fundierten Überblick und schnellen Zugang zum Entwickeln von ontologiebasierten Anwendungen mittels der Protégé-OWL API zu bekommen. Hier ist gegebenenfalls die besser dokumentierte API des Semantic Web Framework Jena 2 für die Entwicklung vorzuziehen. Die Entwicklung von Ontologien ist nicht trivial und sollte gut geplant sein. Die entwickelten Ontologien dienen lediglich als Datenbasis und müssen für eine reale Anwendung reengineered werden. Als Inferenzmaschine kam RacerPro zum Einsatz allerdings lediglich, um die Ontologien und ihre Individuen während der Entwicklung hinsichtlich ihrer Konsistenz zu überprüfen. Für die Anbindung von RacerPro an Protégé wird das DIG-Interface der Description Logic Implementation Group (DIG) eingesetzt. Diese verwendet eine eigene Ontologiesprache, um die Ontologie der Inferenzmaschine zu beschreiben. Bei der Entwicklung der Ontologien ist hierbei aufgefallen, dass OWL und die DIG-Ontologiesprache nicht voll kompatibel sind. Für das OWL `owl:oneOf` Statement gibt es kein entsprechendes Sprachkonstrukt in DIG. Hier soll allerdings demnächst ein neuer Standard DIG 2.0 verabschiedet werden, der dies Problem eventuell behebt. Nähere Informationen zum DIG-Interface lässt sich in (Diederichs, 2005b) oder direkt auf der Homepage der Description Logic Implementation Group (DIG) finden.

7.3 Ausblick

Der Prototyp ist nicht auf dem Stand wie ursprünglich geplant. Zunächst müssten also die weiteren geplanten Meilensteine umgesetzt werden. Um den Prototypen in die SOA Architektur einzubinden müssen entsprechende Web Service Schnittstellen implementiert werden und in den Enterprise Service Bus (ESB) integriert werden. Unerlässlich ist auf lange Sicht die Suche auf Basis einer RDF Query Language umzusetzen. Hier bietet sich die Simple Protocol and RDF Query Language (SPARQLW3C) an. Der Mediator würde dann auf Basis eines Mappings SPARQL-Queries transformieren und an die externen Services schicken. Schließlich könnte man auch die Ontologien um explizite Regeln erweitern. Hierfür könnte man eine Regelsprache wie die Semantic Web Rule Language (SWRL, 2004) verwenden. Generell sollten sowohl die Ontologien, als auch der Prototyp einem gründlichen Redesign unterworfen werden, da einige Komponenten, wie beispielsweise der Mediator momentan nicht vernünftig in eine SOA eingebettet werden können. Für eine echte Demo Anwendung müssten an allen Komponenten (GUI-Frontend, SearchService, Mediator) Erweiterungen vorgenommen werden. Abschließend kann man sagen, dass der Autor im Rahmen des Projektes erste praktische Erfahrungen mit dem Semantic Web gesammelt hat, aber genau wie der Prototyp noch am Anfang steht.

Literatur

[Apache a] APACHE, Software F.: *Apache Struts*. – URL http://struts.apache.org/

[Apache b] APACHE, Software F.: *Apache Tomcat*. – URL http://tomcat.apache.org/

[Diederichs 2005a] DIEDERICHS, Gerrit: *Semantic Web: Enrichment und Search*. Mai 2005a. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2005/diederichs/abstract.pdf

[Diederichs 2005b] DIEDERICHS, Gerrit: *Bausteine für Semantic Web Anwendungen*. Dezember 2005b. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master05-06/diederichs/abstract.pdf

[DIG] DIG: *Description Logic Implementation Group*. – URL http://dl.kr.org/dig/

[Eclipse] ECLIPSE: *Eclipse*. – URL http://www.eclipse.org/

[Gruber 1993] GRUBER, Thomas R.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In: *International Journal Human-Computer Studies* 43 (1993), März, S. 907–928. – URL http://tomgruber.org/writing/onto-design.pdf

[Horrocks] HORROCKS, Ian: *Logical Foundations for the Semantic Web*. – URL http://www.cs.man.ac.uk/~horrocks/Slides/glasgow.pdf

[Jena2] JENA2: *Jena A Semantic Web Framework for Java*. – URL http://jena.sourceforge.net/

[Matthew Horridge 2004] MATTHEW HORRIDGE, Alan Rector Robert Stevens Chris W.: *A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools Edition 1.0*. Online. August 2004. – URL http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf

[Michael C. Daconta 2003] MICHAEL C. DACONTA, Kevin T. S.: *The Semantic Web*. Kap. 5 & 8, Wiley Technology Publishing, 2003

[OWLW3C] OWLW3C: *Web Ontology Language*. – URL http://www.w3.org/2004/OWL/

[Protégé] PROTÉGÉ: *The Protégé Ontology Editor and Knowledge Acquisition System*. – URL http://protege.stanford.edu/

[RACER] RACER: *Racer Systems GmbH & Co. KG*. – URL http://www.racer-systems.com/de/index.phtml

[RDFW3C] RDFW3C: *Resource Description Framework (RDF)*. – URL `http://www.w3.org/RDF/`

[SPARQLW3C] SPARQLW3C: *SPARQL Query Language for RDF*. – URL `http://www.w3.org/TR/rdf-sparql-query/`

[SWRL 2004] SWRL: *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. Mai 2004. – URL `http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/`