Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterprojekt 'Ferienclub' - Master Informatik

Artem Khvat

Semantic Web (Ontologien und Repräsentation des Wissens)

Fakultät Technik und Informatik
Studiendepartment Informatik
Betreuer: Prof. Dr. von Luck
Datum: 15. Februar 2006

Artem Khvat

Thema der Masterprojekt 'Ferienclub' - Master Informatik

Ontologien und Repräsentation des Wissens

Stichworte

Semantic Web, Ontologien, MOnTo, XML, URL, OWL, Jena, Protege

Kurzzusammenfassung

Die vorliegende Ausarbeitung beschäftigt sich mit den Project-Ergebnissen, die im Rahmen der Master Veranstaltung Project "Ferienklub" an der Hochschule für Angewandte Wissenschaften Hamburg, von dem Student Artem Khvat unter der Betreuung des Professors Dr. Kai von Luck erzielt wurden. Sie ist in drei Kapitel aufgeteilt. Im ersten Teil wird das gesamte Vorhaben des Projektes präsentiert, dabei wird kurz auf die einzelnen Teilbereiche des Projektes, dessen Gasamtstruktur und der möglichen Anwendungsfälle sowie Szenarien für den Ferienklub eingegangen. Im zweiten Teil präsentiert der Autor seinen Teilbereich des Projektes, der sich mit den Problemen des Zugriffes auf eine heterogene Umgebung und Integration der neuen Informationsartefakte in schon bestehende Informationssysteme befasst. Der Autor wird die Hauptaspekte des Themas erläutern und eine kurze Einführung in die theoretischen Grundlagen des Themas geben. Im dritten Teil wird das Vorhaben des Autors im Projekt beschrieben. Außerdem werden auch die gewonnenen Erkenntnisse und die dabei auftretenden Schwierigkeiten beschrieben. In diesem Kapitel wird auch die Balance zwischen erreichten Zielen und vorgenommenen Zielen gezogen. Am Ende wird ein Ausblick auf eine bevorstehende Master Thesis, welche sich im Wesentlichen mit den im Projekt behandelten Themen befassen wird, gegeben. Es werden die Hauptziele für Master Thesis und die möglichen Risiken genannt.

Inhaltsverzeichnis

1 Projekt 'Ferienklub'

Im Sommersemester 2005 wurde an der Hochschule für Angewandte Wissenschaften Hamburg im Rahmen des Master Studiengangs ein Projekt mit dem Namen "Ferienklub"gestartet. Die Dauer dieses Projektes wurde auf zwei Semester angesetzt. Das Ziel war es, die Metapher "Ferienklubünter Verwendung der neusten innovativen Technologien aus dem Bereich der Informatik zu implementieren. Die dadurch gewonnenen Erkenntnisse sollten für die Studierende als Einarbeitung in die bevorstehende Masterarbeit dienen und in gewissen Maßen eine Art Risiko- und Machbarkeitsanalyse für die Abschlussarbeit sein.

Im ersten Semester ist von den Studierenden die theoretische Einarbeitung in das jeweils ausgewählte Thema gemacht worden. Die Ergebnisse dieser Arbeit sind auf der Webseite der Hochschule für Angewandte Wissenschaften Hamburg erhältlich [1]. Als Resultat wurden am Ende die wichtigsten Gebiete der Informatik bestimmt, in denen sich die Hauptinteressen der Studierenden befanden. Zusammengefasst waren es dann die folgende Themen:[2]

- Semantic Web
- Persistenz
- Service Oriented Architecture
- RFID
- KI
- 3D-Visualisierung

Es wurden auch die entsprechenden Anwendungsfälle und Szenarien für den Ferienklub entwickelt [3], die Ihre Realisierung durch die oben genannten Techniken finden sollten.

Unter anderen waren es:

- Einchecken des Gastes
- Einführung in die PDA - Bedienung
- Interesseneingabe durch den Gast
- Tagesplaner
- Nutzung von Freizeitangeboten
- etc.

Die Hauptinteressen des Autors lagen im Bereich des Semantic Webs, insbesondere bei den Themen des (Halb-)automatisierten Ontologien-Merges [4], Attribut-Mapping und Transformation, welches von ihm als wichtiges Werkzeug bei der Realisierung des Anwendungsfalls "Nutzung von Freizeitangeboten"[5] angesehen wurde.

Auf Basis der im ersten Semester erarbeiteten theoretischen Grundlagen [6] war der Autor der Meinung, dass die Anwendung von Ontologien eine effiziente Lösung für die vorhandene Heterogenität des Ferienklubs bietet. Diese Annahmen wurden später während der Implementierungs-Phase zum Teil bestätigt. Es wurden aber auch bestimmte Aspekte des Problems gefunden, bei denen das Erreichen der gestellten Ziele wegen noch nicht ausgereifter Techniken oder enormer Komplexität nicht möglich war.

2 Teilprojekt 'Semantic Web'

Um das Problem der Heterogenität im Ferienclub deutlich zu machen, wird hier ein Beispiel für eine Autovermietung im Szenario des Ferienklubs betrachtet. Die ausführliche Beschreibung der notwendigen theoretischen Grundlagen wird in cite006 und cite007 vorgestellt.

Betrachten wir einen Ferienklub, bei dem für die Nutzung von Freizeitangeboten zusätzlich die Möglichkeit einer Autovermietung zur Verfügung steht. Die Autovermittlungsstelle des Ferienclubs muss dabei ihre Vorstellung von dem Auto in irgendeiner Form abspeichern und mit den anderen Autoanbietern abstimmen. Für diese Zwecke bieten sich zwei Lösungsalternativen an.

Im ersten Fall benutzt man eine eigene XML-Form und zwingt alle anderen Anbieter sie zu unterstützen. Das führt dazu, dass alle Autoanbieter die ihr Angebot an den Ferienklub richten wollen, ihre bestehenden XML-Formen für die Autoangebote entsprechend ändern müssen. Das Ganze führt zu einem enormen Verwaltungsaufwand. Eine mögliche Lösung dafür währe die Erstellung eines Transformationsmoduls, was die entsprechenden Attribute der einzelnen XML-Formen einander zuordnet. Jeder Autoanbieter der sein Angebot für den Ferienklub verständlich machen will, muss eine entsprechende Attributmap in dem Transformationsmodul ablegen.

Der Nachteil ist, dass die Erstellung von solchen Attributmaps nur durch den Menschen möglich ist. Grund dafür ist, dass XML selbst keinerlei semantische Informationen beinhaltet. Dieses Thema wird in [7], im Kapitel "Problemäusführlich erläutert. Daraus folgt, dass bei jedem neuen Autoanbieter die Attributmap manuell neu erstellt und registriert werden muss. Die Implementierung dieses Szenarios wurde im Rahmen des Projektes von Thomas Steinberg durchgeführt und ist in seiner Ausarbeitung ausführlich beschrieben.

Eine andere Möglichkeit ist die Verwendung von Ontologien. Man einigt sich auf die Beschreibung des Autoangebotes mittels einer gemeinsamen Ontologiesprache (RDF[8], RDF(S)[9], OWL[][10] etc.). Die eigentliche Modellierung der

Angebotsformulare wird den einzelnen Anbietern selbst überlassen. Wie oben schon erwähnt ist die einzige Voraussetzung die Verwendung einer gemeinsamen Beschreibungsprache. Die Einbettung von Attributmaps und Mergetabellen kann halbautomatisch oder komplett automatisch geschehen. Die semantischen Zusatzinformationen werden dabei für die Automatisierung gebraucht. Der Grad der Automatisierung wird durch eine Einigung auf die Verwendung einer gemeinsamen Upper-Ontologie [11] erhöht. Dadurch kann der Gesamtprozess wesentlich vereinfacht und die manuelle Komponente stark reduzieren.

Offensichtlich ist,dass die vorgeschlagene Lösung viel komplexer in ihrer Realisierung ist, aber im Gegensatz zu der Verwendung von einfachen XML-Formularen bietet sie viel mehr Flexibilität bei der Integration neuer Mitglieder. Außerdem lässt sich durch die semantische Komponente die Konsistenz der Daten auf viel einfachere und elegantere Weise überprüfen. Für diese Zwecke kann man die existierenden Inference-Werkzeuge [12] leicht integrieren und benutzen.

Im Rahmen des Projekts hat sich der Autor entschlossen, den zweiten Ansatz zu realisieren. Dabei ist ein Prototyp mit Namen MOnTo (Merge Ontology Tool) entstanden, welcher die Rolle des semantischen Transformationsmoduls übernehmen sollte. In den folgenden Kapiteln wird dieser Prototyp genauer beschrieben.

2.1 Ziele

Wie oben schon erwähnt hat sich der Autor für die semantische Lösung zur Transformationen von heterogenen Informationen entschlossen. Am Ende der Anforderungsanalyse und der Abstimmung mit den anderen Mitgliedern des Projektes, standen folgende Anforderungen an den Prototypen fest:

- Repräsentation der Ontologien
- Erstellung und Modifikation der Ontologien
- Unterstützung des manuellen Merges von zwei Ontologien
- Unterstützung einer automatischen Anfrageübersetzung zwischen zwei Ontologien (Sprache RDQL [23])
- Untersuchung des automatischen Merge von zwei Ontologien, bei minimaler menschlicher Unterstützung
- Untersuchung von Möglichkeiten zur Vereinfachung des manuellen Merge von Ontologien

Während der Arbeit an dem Projekt sind noch zusätzliche Anforderungen hinzugekommen. Diese Anforderungen waren notwendig für die Integration dieser Teil-Lösung in das Gesamtprojekt. Dabei war die Zusammenarbeit innerhalb der Semantic Web Gruppe [13] vorgesehen. Es kamen folgende Anforderungen hinzu:

- Kommunikation mit der Welt über Web-Services. Dabei sollte es durch den Einsatz der Web-Services möglich sein diese Lösung in eine SOA-Plattform einzubinden.
- Kommunikation mit dem XML-Form Transformationsmodul (Thomas Steinberg). Die Kommunikation mit dem syntaktischen Transformationsmodul gab diesem die Möglichkeit seine syntaktischen Transformationsfähigkeiten zu erweitern, indem es Zugriff auf die Wissens-Datenbank der semantischen Transformation erhält.
- Kommunikation mit einer Service Discovery Engine (Piotr Wendt). Die Suche nach neuen Dienstanbietern passiert viel effizienter, da die Discovery Engine in der Lage ist, die Dienste nach Qualität, Sicherheit und anderen wichtigen Merkmalen zu beurteilen und auszusortieren.

Die meisten von diesen Zielen wurden am Ende des Projekts erreicht. Das Ergebnis ist der Prototyp MOnTo 0.3.

2.2 Werkzeuge

In der Vorbereitungsphase des Projekts wurde eine Untersuchung der möglichen Werkzeuge durchgeführt. Eine sehr wichtige Anforderung war, dass das ausgewählte Werkzeug auf der Programmiersprache Java basieren sollte. Der Grund dafür war die große Erfahrung des Autors mit dieser Sprache. Als Ergebnis dieser systematischen Suche waren die beiden Frameworks Protégé 2000 von der Stanford University [14] (USA) und Jena 2.0 vom HP Research Lab Bristol [15](UK). Die beiden Frameworks sind weit verbreitet und haben sich als mächtige Mittel für die Lösung der unterschiedlichen Aufgaben und Problemstellungen im Bereich des Semantic Webs und insbesondere des Ontologie-Engineerings bewährt. Diese beiden Frameworks werden hier kurz vorgestellt. Dabei wird nicht auf die komplette API sondern nur auf die wesentlichen Merkmale eingegangen.

2.2.1 Protégé

Protégé [16] ist:

- ein Werkzeug um Ontologien zu erstellen und Daten in diese einzutragen
- OpenSource
- eine java-basierte Plattform, die um Plug-Ins erweitert werden kann
- eine Bibliothek die von anderen Anwendungen verwendet werden kann um Daten einzusehen und diese zu manipulieren
- ursprünglich zur Wissensrepräsentation in der medizinischen Informatik von der University of Stanford entwickelt worden

Protégé bietet:

- Erstellung und Manipulation von Ontologien
- Erstellung von Informationen zur weiteren Verwertung
- Lesen, Schreiben und Interkonvertierung folgender Formate:

  Relationale Datenbanken (ODBC)

  CLIPS

  UML / XMI

  XML / XML Schema

  RDF

  Topic Maps

  DAML+OIL

  OWL
- Eine benutzerfreundliche GUI
- Mehrbenutzerfähigkeit
- Ein Protege-Server, mehrere Clients, Wissensdatenbank liegt auf Server, Abgleich mit Clients in Echtzeit
- Echtes verteiltes Arbeiten an einer Wissensdatenbank

Ein wesentlicher Vorteil von Protege ist offensichtlich die Vielfalt der vorhandenen Lösungen, die mit Hilfe von Protege erstellt werden können. Protege bietet ein mächtiges API welches für die Lösung von sehr anspruchsvollen Aufgaben konzipiert wurde. Wie oben schon erwähnt, bietet Protege eine Plattform, die Eclipse [18] sehr ähnlich ist. Diese kann durch Plug-Ins beliebig erweitert werden. Ein Vorteil dabei ist, dass man auf sehr einfache Weise durch geschickte Kombination der Plug-Ins eine sehr effiziente Lösung erschaffen kann, ohne dabei großen Implementierungs-Aufwand betreiben zu müssen. Es gibt aber ein Problem, welches die Arbeit mit Protege, auf Grund des sehr großen Lösungspotenzials, sehr erschwert. Man braucht eine sehr gute Dokumentation um die Einarbeitung möglichst schnell zu ermöglichen. Diese ist bei Protege nicht gegeben. Sorgfältige Untersuchungen des Autors ergaben, dass fast keine Dokumentation für Protege 2000 vorhanden ist. Dadurch wird ein schneller Einstieg verhindert und die Arbeit mit Protege folglich erschwert. Diese Restriktion war für den Autor aufgrund des knappen Zeitrahmens für das Projekt nicht akzeptabel.

2.2.2 Jena

Jena [17] ist ein Java Framework für die Erstellung von Semantic Web Applikationen. Es hat folgende Funktionen:

- RDF API für die Generierung und Verarbeitung von RDF-Dokumenten
- Unterstützung von diversen DB-Systemen z.B. Oracle
- Reasoning Subsystem für RDFS, OWL/Lite und Teile von OWL/Full
- Ontology Subsystem für die Unterstützung bei der Arbeit mit RDFS, OWL, DAML+OIL
- Unterstützung der Sprache RDQL, für die Erstellung von Anfragen an Ontologien

Bei der späteren Arbeit mit Jena 2.0 hat sich herausgestellt, das Jena für manche Arten von Aufgaben nur Grundfunktionalitäten bietet. Dies führt wegen eigener Implementierungsarbeit zu einem Zusatzaufwand bei anspruchvolleren Aufgaben. Ein sehr wichtiger Vorteil bei Jena 2.0 ist die ausführliche Dokumentation mit vielen nützlichen Beispielen und Tutorien. Ein weiterer sehr wichtiger Vorteil von Jena ist die ständige Weiterentwicklung und Verbesserung des Produkts durch das HP Research Lab Bristol [15](UK). Die Entwickler haben große Bereitschaft für Zusammenarbeit gezeigt, und waren sehr sachlich und hilfsbereit bei Supportanfragen des Autors.

Aus diesen Gründen (insbesondere wegen der Dokumentation), hat sich der Autor für den Einsatz von Jena 2.0 als Entwicklungswerkzeug für das Projekt entschieden.

2.3 MOnTo 0.3

Bei MOnTo 0.3 handelt es sich um eine Software für die Arbeit mit Ontologien. Dem Benutzer werden folgende Möglichkeiten durch die Interaktion mit der GUI geboten:

- Repräsentation von Ontologien.
- Erstellung und Modifikation von Ontologien.
- Einfügen, löschen und umbenennen der einzelnen Elemente in Ontologien etc.
- Unterstützung des manuellen Merge's zweier Ontologien.
- Aufdeckung von einfachen Topologisch-syntaktischen Ähnlichkeiten.
- Einbettung der Attributmaps die eine Autoübersetzung zwischen zwei Ontologien erforderlich sind.

- Unterstützung der Ontologiesprachen RDF,RDF(S),OWL.

Die Software wurde, wie schon oben erwähnt mit Hilfe der Jena 2.0 Java API erstellt. Da es sich lediglich um einen Prototypen handelt, sind manche Operationen nicht sehr ergonomisch gestaltet. Dieses Defizit wird in folgenden Versionen von MOnTo beseitigt. Im folgenden Kapitel werden die Gesamtarchitektur und die Implementierung der wichtigsten Funktionalitäten genauer erläutert.

2.3.1 Gesamtarchitektur

Die Gesamtarchitektur besteht aus den drei wesentlichen Blöcken Logik, Controller und GUI welche durch die entsprechenden Java - Klassen realisiert wurden.

2.3.2 Logikblock

Der Logikblock wird in der Klasse MOnToLogik implementiert. Hier sind die Basisalgorithmen- und Operationen für die Arbeit mit Ontologiebäumen implementiert. Durch die geschickte Kombination von diesen Basismethoden ist es möglich viel komplexere Aufgaben effizient zu lösen. Letztere ist die Aufgabe des Controllerblocks.

2.3.3 Controllerblock

Der Controllerblock ist für die Entgegennahme der Nachrichten von der GUI und für die Weiterleitung der Teilaufgaben an den Logikblock zuständig. Die Abarbeitung der GUI - Anfragen wird durch die Komposition von einzelnen Basisoperationen realisiert. Dies erhöht die Wiederverwendbarkeit der einzelnen Module und führt zu einer Zeitersparnis bei der Implementierung neuer Aufgaben.

2.3.4 GUI

Bei der Erstellung der GUI wurde der Standart GUI - Komposer des JBuilder X [20] verwendet. Letzterer ermöglicht die einfache und schnelle Erstellung der notwendigen Benutzeroberflächen. Die gesamte JBuilder X [20] Entwicklungsumgebung ist auf der Webseite von Borland für private und studentische Nutzung frei verfügbar. Es wird lediglich eine einmalige Registrierung mit einer gültigen E - Mail Adresse verlangt. Die meisten Komponenten der GUI kommen aus dem Java Swing Container und werden vom JDK 1.5 unterstützt. Die GUI wird in folgendem Bild [1] dargestellt.

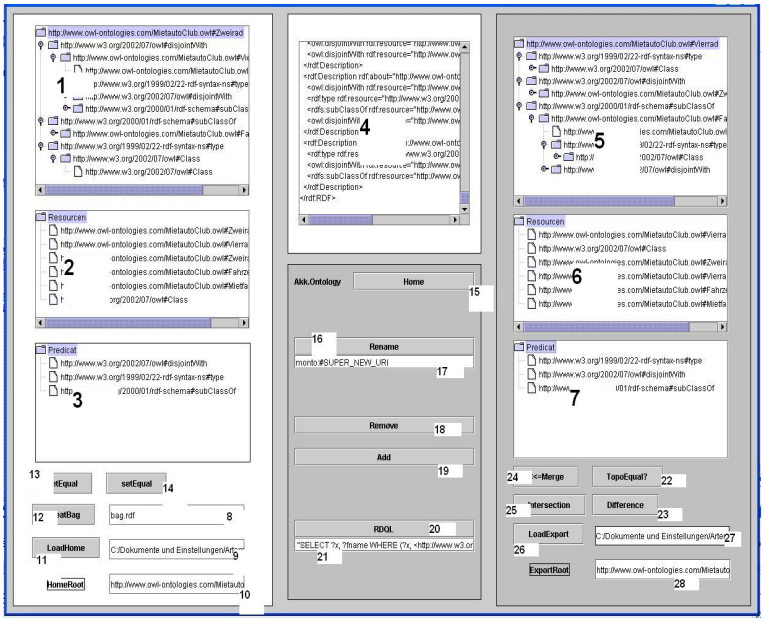

Abbildung 1: GUI-MOnTo

Die GUI ermöglicht die Interaktion mit dem Benutzer. Hier werden nur die wesentlichen Möglichkeiten der Ontologiebearbeitung beschrieben, die dem Benutzer während der Arbeit mit diesem graphischen Interface zur Verfügung stehen. Aber als erstes werden die einzelnen Hauptelemente der Benutzeroberfläche beschrieben. Man hat auf der linken und rechten Seite zwei Gruppen von Anzeigefenstern, die jeweils aus drei Teilen in jeder Gruppe bestehen. Diese Anzeigefenster erlauben die Darstellung der geladenen Ontologien. Das obere Fenster zeigt den gesamten Baum der geladenen Ontologie. Das Fenster in der Mitte jeder Gruppe ist für die Darstellung der Ressourcen der Ontologie. Das Fenster darunter zeigt eine Liste von Prädikaten der geladenen Ontologie. Für die Darstellung der Baumstruktur der Ontologien wurde die Swing Java Klasse JTree [19] benutzt. Dies ermöglicht eine relativ übersichtliche Ansicht der geladenen Ontologie. Der Autor will besonders das Problem der Darstellung von großen Ontologien, was bei der Arbeit an dieser Version der Software zwar erkannt aber nicht gelöst wurde, betonen. Der Autor wird die Lösung dieses Problems in seiner bevorstehenden Masterarbeit zu finden versuchen.

Beschreibung der einzelnen Operationen:

Laden der Ontologien:

- Den Pfad zur gewünschten Ontologie in Eingabefeld [9] oder [27] eingeben.
- Das Rootelement der Ontologie in Eingabefeld [10] oder [28] eingeben.
- Die Taste [11] für die Home-Ontologie oder die Taste [26] für die Export-Ontologie.

Löschen:

- Die gewünschte Ressource oder das Rootelement des Teilbaums in Anzeigefenster [1] oder [5] mit der Maus markieren.
- Mit der Taste [15] die gewünschte Ontologie auswählen.
- Die Taste [18] betätigen.

RDQL Anfrage:

- Mit der Taste [15] die gewünschte Ontologie auswählen.
- Die RDQL-Anfrage in das Eingabefeld[21] eingeben.
- Die Taste [20] betätigen.
- Das Ergebnis wird im Anzeigefenster [4] ausgegeben.

Einfügen:

- Mit der Taste [15] die gewünschte Ontologie auswählen.
- Die gewünschte Ressource aus dem Ressourcen-Anzeigefenster [2] oder [6] auswählen.
- Das gewünschte Prädikat aus dem Prädikat-Anzeigefenster [3] oder [7] auswählen.
- Den gewünschten Teilbaum für das Einfügen in dem Ontologie-Anzeigefenster[1] oder [5] auswählen.
- Die Taste [19] betätigen.

Umbenennen:

- Mit der Taste [15] die gewünschte Ontologie auswählen.
- Die gewünschte Ressource aus dem Ressourcen-Anzeigefenster [2] oder [6] auswählen.
- Den neuen Namen in das Eingabefeld [17] eingeben.
- Die Taste [16] betätigen

Merge:

- Mit der Taste [15] die gewünschte Ontologie auswählen.

- Das Rootelement der gewünschten Teilbäume im Ontologie-Anzeigefenster [1] und [5] auswählen.
- Die Taste [24] betätigen.
- Das Ergebnis des Merges wird im Ontologie-Anzeigefenster [1] ausgegeben.

Überprüfung auf Topologieübereinstimmung:
Bei diesem Prozess werden die beiden Ontologien auf ihre Topologie hin überprüft, wobei die Ressourcen vernachlässigt werden. Es werden nur Ontologiebäume und die Prädikate der jeweiligen Ontologiesprache verglichen. Es wird eine einfache Form der Graphenunifikation durchgeführt, die es erlaubt bei den gleichen Ausgangstopologien die Disjunktheit der Ontologien aufzudecken.

- Das Rootelement der gewünschten Teilbäume im Ontologie-Anzeigefenster [1] und [5] auswählen.
- Die Taste [22] betätigen.
- Das Ergebnis wird in Anzeigefenster [4] ausgegeben

Um diese Operation zu verdeutlichen wird vom Autor folgendes Beispiel vorgeschlagen. Man nimmt die Elemente Vierrad und Zweirad aus der Autoontologie [21] und vergleicht sie. Wenn man die Topologien der Bäume dieser Elemente vergleicht, sind sie absolut identisch, aber durch den Einsatz von Graphenunifikation kann festgestellt werden dass sie disjunkt sind, ohne überhaupt etwas über Zweirad oder Vierrad zu wissen. Es wird möglich, weil die Disjunktheit [22] eine bestimmte topologische Eigenschaft hat welche durch Graphenunifikation feststellbar ist. Der Autor vermutet, dass die Weiterentwicklung dieses Ansatzes eine sehr große Rolle bei der automatischen Integration von neuen Informationsartefakten in bestehende Informationssysteme spielen kann. Der Autor will das Thema weiter in einer Masterarbeit bearbeiten, was wegen der sehr großen Komplexität des Themas nicht im Rahmen dieses Projekts möglich war.
Difference:
Es werden zwei Ontologien verglichen, als Ergebnis kriegt man alle Elemente die in beiden Ontologien unterschiedlich sind.

- Das Rootelement der gewünschten Teilbäume im Ontologie-Anzeigefenster [1] und [5] auswählen.
- Die Taste [23] betätigen.
- Das Ergebnis wird in Anzeigefenster [4] ausgegeben

Intersection:
Es werden zwei Ontologien verglichen, als Ergebnis kriegt man alle Elemente die in beiden Ontologien gleich sind.

- Das Rootelement der gewünschten Teilbäume im Ontologie-Anzeigefenster [1] und [5] auswählen.
- Die Taste [25] betätigen.
- Das Ergebnis wird in Anzeigefenster [4] ausgegeben

Attributmap einlegen:
Eine Attributmap wird angelegt um die Übersetzung zwischen zwei Ontologien zu ermöglichen. In ihr wird die Zuordnung zwischen den Elementen der beiden Ontologien abgespeichert. Es wird dafür ein RDF(S) -Baum aufgebaut und im Hauptverzeichnis als Datei mit dem Namen bag.rdf abgespeichert.

- Den Namen der Datei für die Attributmap in Eingabefeld [8] eingeben.
- Die Taste [12] betätigen.

Attribut in Attributmap speichern:

- 1. Die Ressourcen aus Ressourcen-Anzeigefenster [2] und [6] auswählen
- Die Taste [14] betätigen.

Attribut in Attributmap herausholen::

- Die Ressourcen aus Ressourcen-Anzeigefenster [2] und [6] auswählen
- Die Taste [13] betätigen.
- Das Ergebnis der Operation wird in Anzeigefenster [4] ausgegeben.

2.4 Ergebnisse

Am Ende des Projektes hat der Autor folgende Ziele erreicht:

- Repräsentation der Ontologien.
- Erstellung und Modifikation der Ontologien.
- Unterstützung des manuellen Merge's von zwei Ontologien.
- Unterstützung der automatischen Anfragenübersetzung zwischen zwei Ontologien (Sprache RDQL [23]).
- Untersuchung des automatischen Merge von zwei Ontologien, mit minimaler menschlicher Unterstützung.
- Untersuchung von Vereinfachungsmöglichkeiten für den manuellen Merge von zwei Ontologien.

Als unerreicht blieben die folgenden am Anfang des Projektes gestellten Aufgaben:

- Kommunikation mit der Welt über Web-Services.
- Kommunikation mit dem XML-Form Transformationsmodul
- Kommunikation mit der Service Discovery Engine

Bei den Punkten 1 und 2 war der Zeitaspekt der Grund zum scheitern, der Autor hat den notwendigen Zeitaufwand für die Realisierung dieser Aufgaben falsch eingeschätzt. Bei dem Punkt drei war das Problem, das die notwendige Lösung gar nicht vorhanden war. Die Teilgruppe aus dem Bereich des Semantic Webs ist bei der Implementierung auf Probleme gestoßen, die die Entwicklung einer Service Discovery Engine verhinderten. Die genaue Beschreibung dieses Teilprojektes kann aus der Ausarbeitung von Piotr Wendt entnommen werden.

Als Fazit kann man das Projekt insgesamt als erfolgreich betrachten, und die dabei entstandene Software als nutzbare Basis für die weitere Arbeit im diesem Gebiet verwenden. Es wurde noch einmal gezeigt, dass die automatische Anpassung von neuen semantisch annotierten Informationsartefakten in ein bestehendes System sehr schwer und fast unmöglich zu realisieren ist. Der Mensch wird immer als Hauptakteur bei diesem Prozess benötigt. Die Frage ist nur in wie weit lässt sich seine Rolle minimieren und durch welche Techniken wird dies möglich. Diese zwei Hauptthemen will der Autor in seiner bevorstehenden Masterarbeit intensiv behandeln.

3 Master Thesis

Die Master Thesis wird auf den Ergebnissen des Projekts aufbauen. Die Software MOnTo wird weiter entwickelt und verbessert. Die folgenden Themen werden dabei als Kernthemen behandelt:

- Bestimmung der Grenzen des Automatisierens der Integration.
- Die Weiterentwicklung der notwendigen Techniken für die Automatisierung der Integration. Die Graphenunifikation und einfacher topologischer Vergleich werden dabei als wichtige Techniken betrachtet.
- Fragen der Ergonomie und Vereinfachung des Merge's von großen Ontologien.

Das größte Risiko bei dieser Master Thesis sieht der Autor im Zeitaspekt und der Komplexität des Themas. Es ist offensichtlich, dass die angestrebten Aufgaben nicht in der dafür vorgesehenen Zeit von 6 Monaten komplett zu schaffen sind. Aus diesem Grund ist es sehr wichtig die Ziele klar abzugrenzen und bestimmte

Bereiche eventuell offen zu lassen. Die exakte Definition der Ziele und die dazu notwendige Zeit lassen sich auf Basis der im Projekt gewonnenen Erkenntnisse leichter und exakter durchführen.

Literatur

[1] http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2005/vortraege.html

[2] http://www.informatik.haw-hamburg.de/wiki/inf-master/index.php/Anwendungen_I_%28AI%29

[3] http://www.informatik.haw-hamburg.de/wiki/inf-master/index.php/Anwendungsf%C3%A4lle_und_Szenarien_f%C3%BCr_den_Ferienklub

[4] http://www.informatik.haw-hamburg.de/wiki/inf-master/index.php/Anwendungen_Thema:_Semantic_Web

[5] http://www.informatik.haw-hamburg.de/wiki/inf-master/index.php/Anwendungsf%C3%A4lle_und_Szenarien_f%C3%BCr_den_Ferienklub#Nutzung_von_Freizeitangeboten

[6] http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2005/khvat/abstract.pdf

[7] http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master05-06/khvat/abstract.pdf

[8] http://www.w3.org/RDF/

[9] http://www.w3.org/TR/rdf-schema/

[10] http://www.w3.org/TR/owl-features/

[11] Asucion Gomez-Perez Mariano Fernadez-Lopez, Oscar Corcho Ontological Engineering Springer Verlag 2003

[12] http://jena.sourceforge.net/inference/index.html

[13] http://www.informatik.haw-hamburg.de/wiki/inf-master/index.php/Anwendungen_Thema:_Semantic_Web

[14] http://www.stanford.edu/

[15] http://www.hpl.hp.com/semweb/index.htm

[16] http://protege.stanford.edu/

[17] http://www.hpl.hp.com/semweb/jena.htm

[18] http://www.eclipse.org/

[19] http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JTree.html

[20] http://www.borland.com/us/products/jbuilder/index.html

[21] https://users.informatik.haw-hamburg.de/home/pub/k/khvat_a/MietAutoClubBkUp.txt

[22] http://de.wikipedia.org/wiki/Lineare_Disjunktheit

[23] http://www.w3.org/Submission/RDQL/