Web Knowledge Management

Seminar: Business Intelligence –
Teil II: Data Mining & Knowledge Discovery

von Anne Jannasch

Betreuer: Dipl. Inform. Marcus Flehmig

13.02.2004

Inhalt

1 Einleitung

Das Internet hat sich in den letzten Jahren zu einem Massenmedium entwickelt und ist zu einer sehr großen Quelle für Informationen aller Art angewachsen. Ebenso ist in Unternehmen ein zunehmendes Wachstum an häufig unstrukturierten Daten und Informationen zu beobachten. Die Datenflut im Internet und in Unternehmen ist enorm und dieser Trend bleibt auch weiterhin bestehen. So ist es nicht verwunderlich, dass sich ein Nutzer nur schwer einen Überblick über die existierenden Angebote an Informationen verschaffen kann, die er nutzen möchte. Es ist eine große Herausforderung, diese unstrukturierte Datenflut in relevante Information umzuwandeln, indem der Benutzer bei der Auswertung der Daten unterstützt wird, damit er die von ihm gewünschten Informationen erhält. Dieses Ziel verfolgend befasst sich Web Wissensmanagement mit dem gezielten Umgang von Wissen in Firmen-Intranets sowie im Internet. Darunter fallen Methoden zur Verbesserung der Erzeugung, Verteilung, des Zugriffs und der Nutzung von Wissen. Die Verwendung von Ontologien wird hierbei eine entscheidende Rolle spielen.
Das zweite Kapitel beschäftigt sich mit dem Begriff der Ontologien, sowie mit Ontologiesprachen. Ontologiewerkzeuge, die bei der Entwicklung von Ontologien hilfreich sind, bilden das Thema des dritten Kapitels. In Kapitel 4 werden zwei Ontologie-basierte Systeme für Wissensmanagement vorgestellt. Das fünfte Kapitel befasst sich mit Methoden zur Extraktion und Verwendung von taxonomischen Relationen.

2 Ontologien und Ontologiesprachen

2.1 Der Ontologie-Begriff

Die Anzahl der gespeicherten Informationsquellen und der verschiedenen Formate dieser Informationen steigt immer weiter an. Dies erschwert das Finden, den Zugriff und auch die Zusammenfassung von Informationen zu einem bestimmten Themenbereich. Um diese Probleme anzugehen, spielt die Verwendung von Ontologien eine wichtige Rolle.
Es gibt viele verschiedene Definitionsmöglichkeiten für den Ontologie-Begriff. Eine häufig verwendete Beschreibung stammt von Tom Gruber[1]: *„An Ontology is a specification of a conceptualization“*, was übersetzt werden kann mit: Eine Ontologie ist eine Spezifikation einer Konzeptualisierung. Unter einer Konzeptualisierung ist eine abstrakte, vereinfachte und formalisierte Ansicht der Welt oder eines Wissensbereichs zu verstehen, die repräsentiert werden soll. Weiterhin soll das Wissen, das die Ontologie beinhaltet, nicht auf ein Individuum eingeschränkt sein, sondern von Gruppen akzeptiert werden. Die Konzeptualisierung einer Ontologie zeichnet sich durch folgende Eigenschaften aus:

- Intensionale Charakterisierung von Konzepten und Beziehungen zwischen den Konzepten, die für einen Wissensbereich als relevant erachtet werden. Ein Konzept wiederum ist eine begriffliche Beschreibung eines Sachverhalts und kann durch Attribute detailliert beschrieben werden.
- Die Konzept- und Beziehungsdefinitionen können durch Einschränkungen und Regeln ergänzt werden.
- Die Ontologie kann durch einen Formalismus beschrieben werden.

[1] http://www-ksl.stanford.edu/kst/what-is-an-ontology.html

Zusammengefasst kann gesagt werden, dass eine Ontologie im Wesentlichen aus Konzepten, Eigenschaften, Beziehungen und Einschränkungen bzw. Regeln besteht.
Durch die Verwendung von Regeln lassen sich logische Schlussfolgerungsketten aufbauen und somit implizite Verknüpfungen erkennen. Dadurch wird neues Wissen abgeleitet, also die Möglichkeit zu einer Inferenz gebildet.
Die Organisation einer Ontologie basiert auf Taxonomien, d.h. die Konzepte sind hierarchisch angeordnet. Von den Konzepten einer Ontologie können Instanzen gebildet werden. Instanzen sind alle Objekte, die durch ein Konzept beschrieben werden. Zusammen mit der Ontologie bilden die Instanzen eine Wissensbasis.

Der Funktion einer Ontologie im Web Wissensmanagement liegt letztendlich darin, Wissen über einen bestimmten Bereich einheitlich zu repräsentieren und neues Wissen abzuleiten.

2.2 Ontologiesprachen

Ontologien lassen sich mit Hilfe von Ontologiesprachen formal beschreiben. Mit einer Ontologiesprache können Konzepte, Eigenschaften von Konzepten und Beziehungen zwischen Konzepten sowie weitere Regeln beschrieben werden. Ontologiesprachen erlauben es, Ressourcen im Web zu annotieren, d.h. zusätzliche Informationen über die Semantik der vorhandenen Informationen beizufügen. Konkrete Annotierungen damit führen zu so genannten Metadaten, also Daten über Daten. Bekannte Ontologiesprachen sind RDF mit RDFS, OIL und F-Logic.

2.2.1 RDF und RDFS

Ontologien können mit Hilfe von RDF[2] (Ressource Description Framework) in Verbindung mit RDFS[3] (RDF-Schema) abgebildet werden.
RDF ist ein Modell zur Repräsentation von Metadaten und wird in einer W3C Empfehlung spezifiziert. Das RDF-Datenmodell besteht aus den drei grundlegenden Objekttypen Ressource, Eigenschaft und Aussage.

- **Ressourcen** („Resources“) sind alle Dinge, die durch RDF beschrieben werden können. Sie werden durch eine URI (Uniform Resource Identifier) identifiziert. Es kann sich dabei sowohl um eine Internet-Seite als auch um ein spezielles HTML-Element oder um einen Gegenstand außerhalb des Internets handeln.
- **Eigenschaften** („Properties“) beschreiben Charakteristiken, Attribute oder Relationen einer Ressource. Auch Eigenschaften sind Ressourcen und können somit wiederum beschrieben werden.
- **Aussagen** („Statements“) bestehen aus drei Teilen: Subjekt, Prädikat und Objekt. Das Subjekt ist eine Ressource, über die die Aussage gemacht wird, das Prädikat ist eine bestimmte Eigenschaft und das Objekt der Wert dieser Eigenschaft. Das Objekt einer Aussage kann eine Ressource sein oder ein Literal.

[2] http://www.w3.org/TR/1999/PR-rdf-syntax-19990105/
[3] http://www.w3.org/TR/rdf-schema/

Beispiel: Folgender Satz soll in RDF repräsentiert werden:

Die Ressource http://www.w3.org/Home/Lassila hat den Autor Ora Lassila

Die Bestandteile der Aussage sind:

Subjekt (Ressource)	http://www.w3.org/Home/Lassila
Prädikat (Eigenschaft)	Autor
Objekt (Wert)	“Ora Lassila”

Das RDF-Datenmodell bietet also ein abstraktes Hilfsmittel, um Aussagen darzustellen. Um solche Aussagen tatsächlich im Web verwenden zu können, braucht es eine Syntax, damit diese Metadaten auch erstellt und ausgetauscht werden können. Zu diesem Zweck kann RDF die XML Syntax verwenden.

In XML-Syntax sieht das obige Beispiel wie folgt aus:

```
(1)<?xml version="1.0"?>
(2)  <rdf:RDF
(3)    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
(4)    xmlns:s="http://description.org/schema/">
(5)      <rdf:Description about="http://www.w3.org/Home/Lassila">
(6)        <s:Creator>Ora Lassila</s:Creator>
(7)      </rdf:Description>
(8)  </rdf:RDF>
```

In den Zeilen 3 und 4 werden zwei Namensräume deklariert: dem RDF-Standardnamensraum wird der Präfix „rdf“ und einem Schema der Präfix „s“ zugewiesen. In Zeile 5 beginnt die eigentliche Beschreibung mit dem Abschnitt „Description“. Über das Attribut „about“ wird auf die Ressource "http://www.w3.org/Home/Lassila" Bezug genommen und damit das Subjekt der Aussage festgelegt. Zeile 6 enthält die Eigenschaft und ihren Wert (Objekt). Die Eigenschaft wird über das Namensraum-Präfix „s“ einem eindeutigen Vokabular zugewiesen.

Mir RDF werden nur Aussagen auf Instanzebene gemacht. Es ist nicht möglich zu beschreiben, welche Typen von Ressourcen es gibt und welche Eigenschaften sie besitzen können. Diesem Nachteil tritt RDF-Schema, eine Erweiterung von RDF, mit der Einführung der formalen Definition von Klassen und zugehörigen Eigenschaften entgegen. Zudem ist es möglich, Vererbungshierarchien für Klassen und Einschränkungen für Eigenschaften zu erzeugen. RDF-Schema bietet also die Möglichkeit Schemata zu beschreiben, die mit Ressourcen instanziiert werden.

Die vordefinierten Klassen des RDFS-Sprachkerns sind:

- **rdfs:Resource** - Alle Einheiten (Ressourcen), die im Zusammenhang mit RDF Ausdrücken verwendet werden, sind Instanzen dieser Klasse.
- **rdf:Property** - Repräsentant der Teilmenge der Ressourcen, die Eigenschaften darstellen.
- **rdfs:Class** - Analog zu Klassen objektorientierter Systeme in dem Sinne, dass jedes neue Konzept eine Eigenschaft rdf:type vorweisen muss, dessen Wert die Ressource rdfs:Class ist.

Folgende Eigenschaften sind in RDFS vorgesehen:

- **rdf:type -** Wird verwendet, um auszudrücken, dass eine Ressource Instanz einer bestimmten Klasse ist. Dabei kann eine Ressource Instanz mehrerer Klassen sein.
- **rdfs:subClassOf -** Beschreibung einer Vererbungsbeziehung zwischen je zwei Klassen. Diese Beziehung ist transitiv und es ist Mehrfachvererbung erlaubt.
- **rdfs:subPropertyOf** - Beschreibung einer Vererbungsbeziehung zwischen Properties Natürlich ist auch die rdfs:subPropertyOf-Relation transitiv.

Das Erstellen einer Ontologie in RDF bedeutet nun, dass ein RDF-Schema definiert wird, welches alle Konzepte (Klassen) und Beziehungen zwischen diesen Konzepten eines Bereichs beschreibt. Um ein RDF-Schema in einem RDF-Dokument zu verwenden, wird das RDF-Schema in einem Namensraum definiert und das Dokument mit diesem Namensraum verbunden.

2.2.2 OIL

Die Sprache OIL[4] (Ontology Inference Layer) wurde für die Repräsentation von Ontologien im Rahmen des On-To-Knowledge Projektes (siehe Kapitel 4.2) entwickelt. Sie führt drei Sprachfamilien zusammen, nämlich Frame-basierte Systeme, Deskriptive Logik und Web-Sprachen (siehe [GRÄB03]).

- **Frame-basierte Systeme** bieten zur Modellierung die zentralen Elemente Frames (Klassen) und Slots (Attribute). Die Frames bilden eine Klassenhierarchie, Slots können mit zusätzlichen Einschränkungen (Restriktionen) versehen werden. Die meisten Frame-basierten Systeme bieten darüber hinaus zusätzliche Modellierungsmöglichkeiten. Viele Aspekte Frame-basierter Systeme finden sich in der objektorientierten Welt wieder. Auch OIL verwendet die grundlegenden Primitiven Frame-basierter Systeme in seiner Sprache. OIL basiert ebenfalls auf dem Konzept der Klassen und der Definition von Superklassen und Attributen. Relationen können dabei auch als eigenständige Entitäten definiert werden, so dass auch sie, wie Klassen, eigene Attribute (z.B. Domain und Range) haben und in einer Hierarchie angeordnet werden können.
- **Deskriptive Logik** (DL) ist eine weitere Methode der Wissensrepräsentation. Sie beschreibt Wissen mittels Konzepten und Rollen (ähnlich den Frames und Slots in Frame-basierten Systemen). Die Semantik von Ausdrücken der DL kann mathematisch präzise beschrieben werden, wodurch beispielsweise Inferenzen möglich sind. OIL übernimmt diese formale Semantik aus der DL und damit auch die Unterstützung für Inferenzen.
- **Web-Sprachen** werden gebraucht, um die OIL Syntax mit gängigen W3C-Standards wie XML und RDF kompatibel zu machen. OIL ist eine Erweiterung von RDF und RDFS. Der Einbezug von Websprachen war ein wichtiger Schritt, da Ontologien im Internet häufig Verwendung finden.

[4] http://www.ontoknowledge.org/oil/

Die Schichten von OIL

OIL ist in Schichten eingeteilt (siehe Abbildung 1). Die unterste Schicht ist Core OIL, dann kommt Standard OIL, danach Instance OIL und zum Schluss Heavy OIL. Core OIL stimmt größtenteils mit dem RDF Schema überein. Agenten, die auf dem RDF Schema arbeiten, können dementsprechend auf Core OIL arbeiten. Standard OIL beinhaltet die grundlegenden Modellierungsprimitiven und stellt den Kern der Sprache OIL dar. Instance OIL beinhaltet die Integration von Instanzen. Heavy OIL reichert die vorhergehenden Schichten um weitere Repräsentations- und Inferenzmechanismen an. Diese Schicht wird in Zusammenarbeit mit der DAML[5] Initiative gebildet.

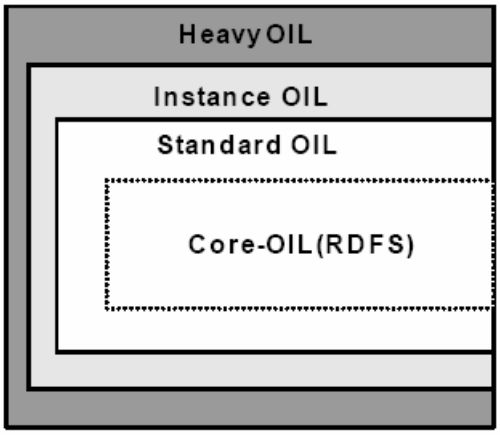

Abbildung 1: Die OIL-Architektur

Der Vorteil der Schichten-Architektur von OIL liegt auf der Hand. Anwendungsentwickler können jene Schicht verwenden, die für ihre Anforderungen genügend Ausdrucksstärke bietet. Die jeweils höhere Schicht ist reicher an Funktionalität und Komplexität, als die darunterliegenden Schichten. Anwendungen, die nur Ausdrücke einer niedrigen Schicht verarbeiten können, wählen nur jene Informationen, die für sie von Nutzen sind. Anwendungen, die einer höheren Komplexität von OIL angepasst werden, verstehen auch Informationen, die durch die niedrigeren Schichten dargestellt werden können.

2.2.3 F-Logic

F-Logic ist eine Ontologiesprache und stellt eine Kombination der Repräsentationsformalismen Frames und Prädikatenlogik dar. Die Anlehnung an prädikatenlogische Spezifikationen findet sich unter anderem in der Möglichkeit zur Definition von Inferenzregeln wieder. Somit kann neues Wissen in einer ontologie-basierten Wissensbasis explizit gemacht werden. Diese Fähigkeit macht sich das Ontologie-basierte System Ontobroker mit der Verwendung von F-Logic zu Nutzen. Für weitere Informationen zur Syntax von Ontobroker sei auf [ONTO] verwiesen.

3 Ontologiewerkzeuge

Wie im zweiten Kapitel dargelegt, werden Ontologiesprachen zur formalen Erstellung von Ontologien verwendet. Der Arbeitsaufwand wäre jedoch enorm, wenn bei der Ontologieerstellung jede einzelne Zeile von Hand geschrieben werden müsste. Der Aufbau einer Ontologie muss also durch kompetente Hilfe unterstützt werden. Aus diesem Grund wurde in den letzten Jahren eine Reihe von Ontologiewerkzeugen erstellt, die bei der Entwicklung und Verwendung von Ontologien hilfreich sind. Die einzelnen Ontologiewerkzeuge unterstützen jeweils verschiedene Funktionalitäten wie beispielsweise das Erstellen, das Visualisieren, die Auswertung, die Überprüfung oder das Verschmelzen von Ontologien.

Im Folgenden wird mit Protégé-2000[6] ein mächtiger Ontologie-Editor vorgestellt, der im Internet kostenlos zur Verfügung steht. Es handelt sich dabei um ein Open-Source-Projekt der Universität Stanford. Protégé-2000 wurde in Java geschrieben und kann durch zahlreiche Plugins, die auf der Homepage zu finden sind, erweitert werden.

Mit Protégé-2000 kann man Ontologien erstellen und bearbeiten. Das Wissensmodell von Protégé-2000 ist Frame-basiert. Eine Protégé-Ontologie besteht aus Klassen, Slots, Facetten und Axiomen.

[5] http://www.daml.org
[6] http://protege.stanford.edu/index.html

Klassen sind Konzepte aus dem zu modellierenden Bereich. Slots entsprechen den Attributen dieser Klassen. Facetten beschreiben Eigenschaften von Slots. Axiome spezifizieren zusätzliche Einschränkungen. Eine in Protégé-2000 erstellte Wissensbasis enthält die Ontologie sowie individuelle Instanzen von Klassen mit deren bestimmten Attributwerten (siehe [CHAO]).

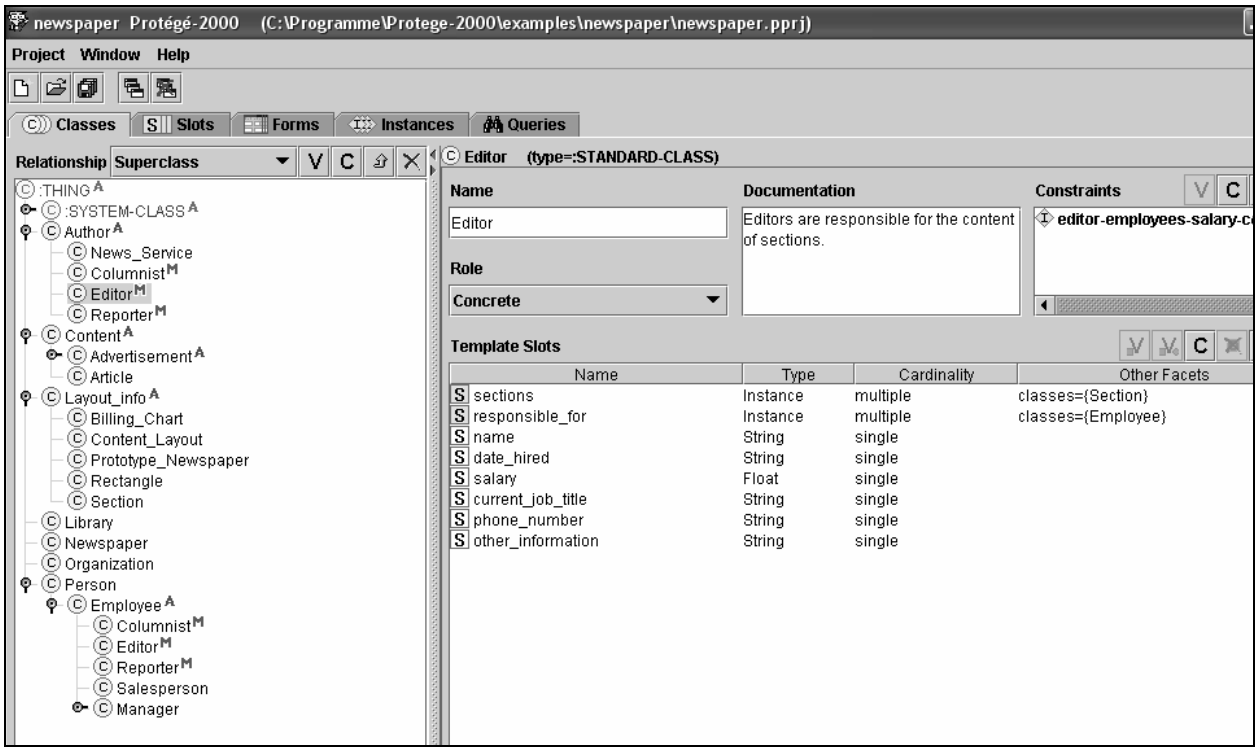

Abbildung 2: Screenshot von Protégé-2000 - Darstellung der Ontologie „newspaper“

Abbildung 2 zeigt die graphische Oberfläche von Protégé-2000. Es sind verschiedene überlappende Tabs zu sehen (Classes-, Slots-, Forms-, Instances- und Queries-Tab). Auf der linken Seite der Abbildung 2 ist der Baum der Klassenhierarchie zu sehen, rechts sind die Slots der im Baum selektierten Klasse (hier: die Klasse „Editor“). Die Wurzel der Klassenhierarchie in Protégé-2000 ist die Metaklasse „THING“. Eine Metaklasse ist eine Klasse, deren Instanzen Klassen sind.
Ist eine Klasse A eine Unterklasse einer Klasse B so gilt für jede Instanz der Klasse A auch, dass sie eine Instanz der Klasse B ist. Protégé-2000 unterstützt die Mehrfachvererbung, d.h. zwei Klassen können eine gemeinsame Unterklasse haben. Diese Unterklasse (in Abbildung 2 mit **M** gekennzeichnet) erbt die Eigenschaften beider Superklassen.
Klassen können konkret oder abstrakt sein. Konkrete Klassen können direkte Instanzen haben, abstrakte Klassen (in Abbildung 2 mit **A** gekennzeichnet) nicht.
Slots entsprechen den Attributen. Sie beschreiben die Eigenschaften der Klassen und Instanzen, wie z.B. der Name oder das Gehalt eines Mitarbeiters. In Protégé-2000 sind Slots als Frames definiert. Sie werden unabhängig von den Klassen erstellt. Wird ein Slot einem Frame in der Benutzer-Ontologie zugeordnet, so beschreibt er die Eigenschaften dieses bestimmten Frames.
Facetten beschreiben Eigenschaften von Slots. Sie definieren Beschränkungen (Constraints) auf der Zuordnung eines Slots zu einer bestimmten Klasse. In Protégé-2000 können folgende Constraints durch Facetten spezifiziert werden (siehe Abbildung 3).

- Festlegen des Werttyps (Integer, String, Instanz einer Klasse, Float etc.),
- Kardinalität eines Slots,
- Minimaler und maximaler Wert eines numerischen Slots, usw.

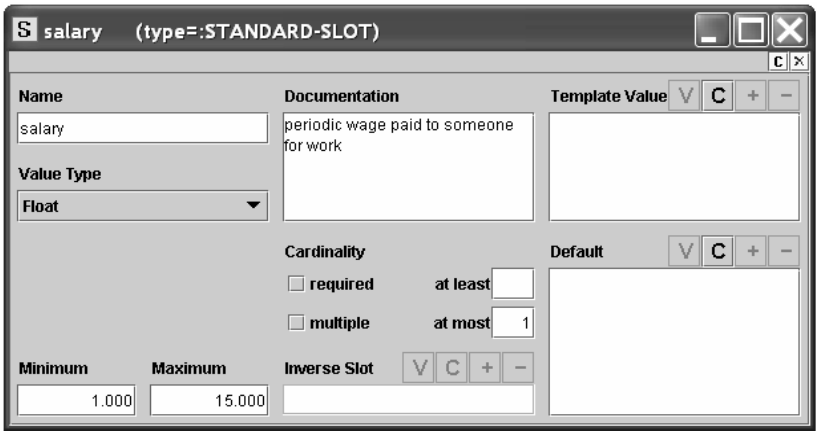

Abbildung 3: Frame des Slots „salary

Im „Forms"-Fenster kann man sich Prototypen von Eingabemasken anschauen und editieren. Diese erstellten Eingabemasken können dann in ihrem endgültigen Format im „Instances" – Fenster eingesehen werden und dienen dort zum Erfassen des Wissens.
Im „Query"-Fenster werden Anfragen eingegeben. Dabei kann die Klasse der gesuchten Instanz, sowie die erforderliche Eigenschaft und einer bestimmten Bedingung angegeben werden. Durch Drücken des Buttons „Find" werden die Suchergebnisse angegeben (siehe [LESE03]).

Die erstellten Ontologien können in verschiedenen Formaten wie beispielsweise RDF-Schema importiert und exportiert werden. Die gesamte Wissensbasis kann durch JDBC-Kompatibilität auch in MSSQL, MySQL, Oracle und MS Access gespeichert werden. Des Weiteren gibt es für Protégé-2000 verschiedene Plugins, die man in drei Kategorien einteilen kann.

- Die Backends bieten die Funktionalität Ontologien in verschiedenen Formaten abzuspeichern und einzulesen.
- Die Tab Plugins, sie stellen eine Schnittstelle oder eine Sicht auf eine Wissensbasis dar. Es kann mehrere Gründe geben, sie zu installieren oder zu entwickeln. Beim Aufbau von Wissensbasen über spezielle Anwendungsbereiche, braucht man vielleicht konkrete Methoden um Wissen zu modellieren, die für diesen Bereich spezifisch sind.
- Die Slot Widgets sind da, um Werte von Slots sichtbar zu machen oder zu editieren, die mit Standard Methoden nicht in eine Wissensbasis eingegliedert werden können, wie z.B. Bilder im JPG oder GIF-Format.

Protégé-2000 wird von Systementwicklern und Experten eines Anwendungsbereichs benutzt, um wissensbasierte Systeme zu entwickeln. Derzeit unterstützt Protégé-2000 mit anderen Anwendungen Problemlösungs- und Entscheidungsprozesse im Anwendungsfeld der Medizin und Biologie.

4 Ontologie-basierte Systeme

Wissensmanagement hat sich in den vergangenen Jahren zu einem kritischen Erfolgsfaktor für Unternehmen entwickelt. Die Globalisierung der Märkte, das Entstehen virtueller Unternehmen, die stärkere Kundenorientierung oder die zunehmende Komplexität von Produkten sind einige der Gründe, weshalb das systematische und gezielte Managen von Wissen immer mehr an Bedeutung gewinnt. Viele Dokument-Management-Systeme haben noch gravierende Schwächen, beispielweise bei der Informationssuche und der Wartung von großen und schlecht strukturierten

Informationsquellen. Die Wettbewerbsfähigkeit vieler Unternehmen hängt stark davon ab, wie sie ihr gemeinsames Wissen ausnutzen können (siehe [FHD+03]).
Um diese Probleme anzugehen, wurden Ontologie-basierte Systeme entwickelt. Ontologie-basierte Systeme unterscheiden sich von anderen Systemen, die zum Wissensmanagement genutzt werden, dadurch, dass die sie Fähigkeit besitzen, Wissen mit Hilfe von Ontologien zu verbinden und neues Wissen abzuleiten. Im Folgenden werden Ontobroker und das On-To-Knowledge Projekt, zwei Ontologie-basierte Systeme, vorgestellt.

4.1 Ontobroker

Ontobroker [SCHW02] ist ein Wissensmanagement-System, das auf Ontologien basiert und von der Firma Ontoprise[7] in Karlsruhe entwickelt wurde. Es ermöglicht den Zugriff auf verschiedenartig strukturierte Quellen und erlaubt das Ableiten von neuem Wissen aus vorhandenen Fakten.

4.1.1 Aufbau

Abbildung 4 beschreibt den Aufbau von Ontobroker. Das System bestehen aus vier Kernmodulen, nämlich dem Info Agenten, der Inference Engine, der Query Engine und dem DB-Manager. Der zentrale Bestandteil sind die Ontologien. Sie werden von verschiedenen Komponenten des Systems verwendet und in einer Repräsentationssprache dargestellt, die auf F-Logic basiert und somit die Möglichkeit bietet, aus vorhandenem Wissen neues Wissen abzuleiten.

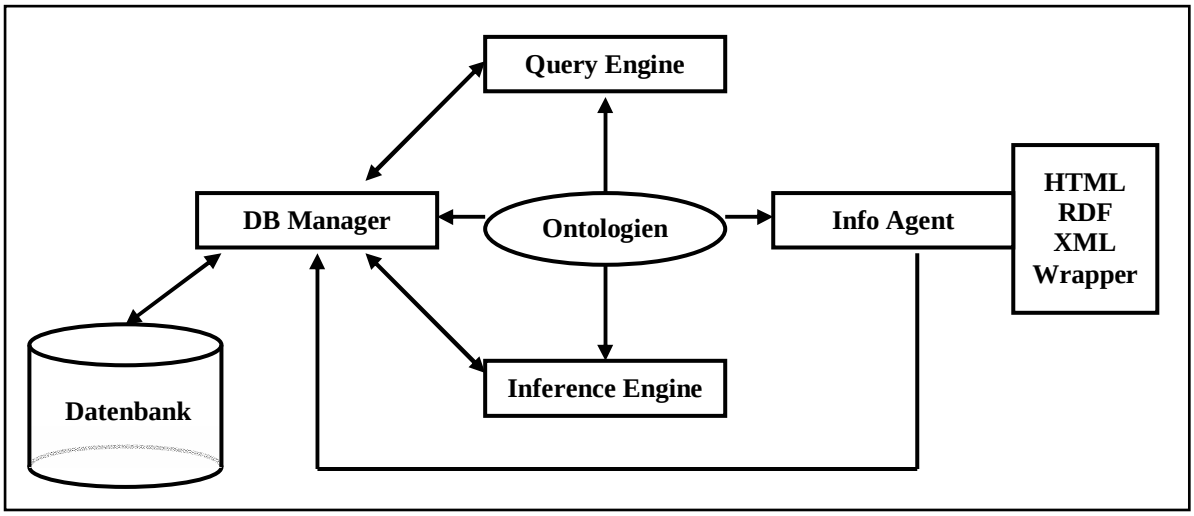

Abbildung 4: Aufbau von Ontobroker

- **Info Agent** - Der Info Agent holt in bestimmten Zeitabständen Informationen aus dem Internet und speichert diese in der Datenbank. Da Ontobroker auf Ontologien basiert, kann der Info Agent dabei sowohl auf strukturierte, semistrukturierte als auch unstrukturierte Quellen zurückgreifen.
  Informationen, die in strukturierter Form vorliegen, sind z. B. Informationen, die aus Datenbanken generiert wurden. Auf diese kann mit Hilfe eines Wrapper-Mechanismus zugegriffen werden, der die Inhalte nach einem vorgegebenen Muster durchsucht. Ontobroker ist somit in der Lage, fest strukturierte Informationen auszuwerten, ohne dass diese erst annotiert werden müssen.
  Semistrukturierte Quellen kommen beispielsweise in Form von XML oder RDF-Dateien vor. Diese erlauben eine freiere Darstellung von Informationen. Es muss kein Standardseitenlayout verwendet werden. Jede Seite kann ein individuelles Layout haben.

[7] http://www.ontoprise.de

Dies ist möglich, da XML und RDF zusätzlich maschinenlesbare Meta-Informationen zur Verfügung stellen.
Bei unstrukturierten Quellen handelt es sich um HTML-Seiten ohne zusätzliche maschinenlesbare Informationen zum Inhalt. Wenn man diese Texte annotiert, ist Ontobroker in der Lage, die vorhandenen Informationen zu verarbeiten. Bei dieser Annotation wird der HTML „anchor“-Tag um das „onto“-Attribut erweitert.

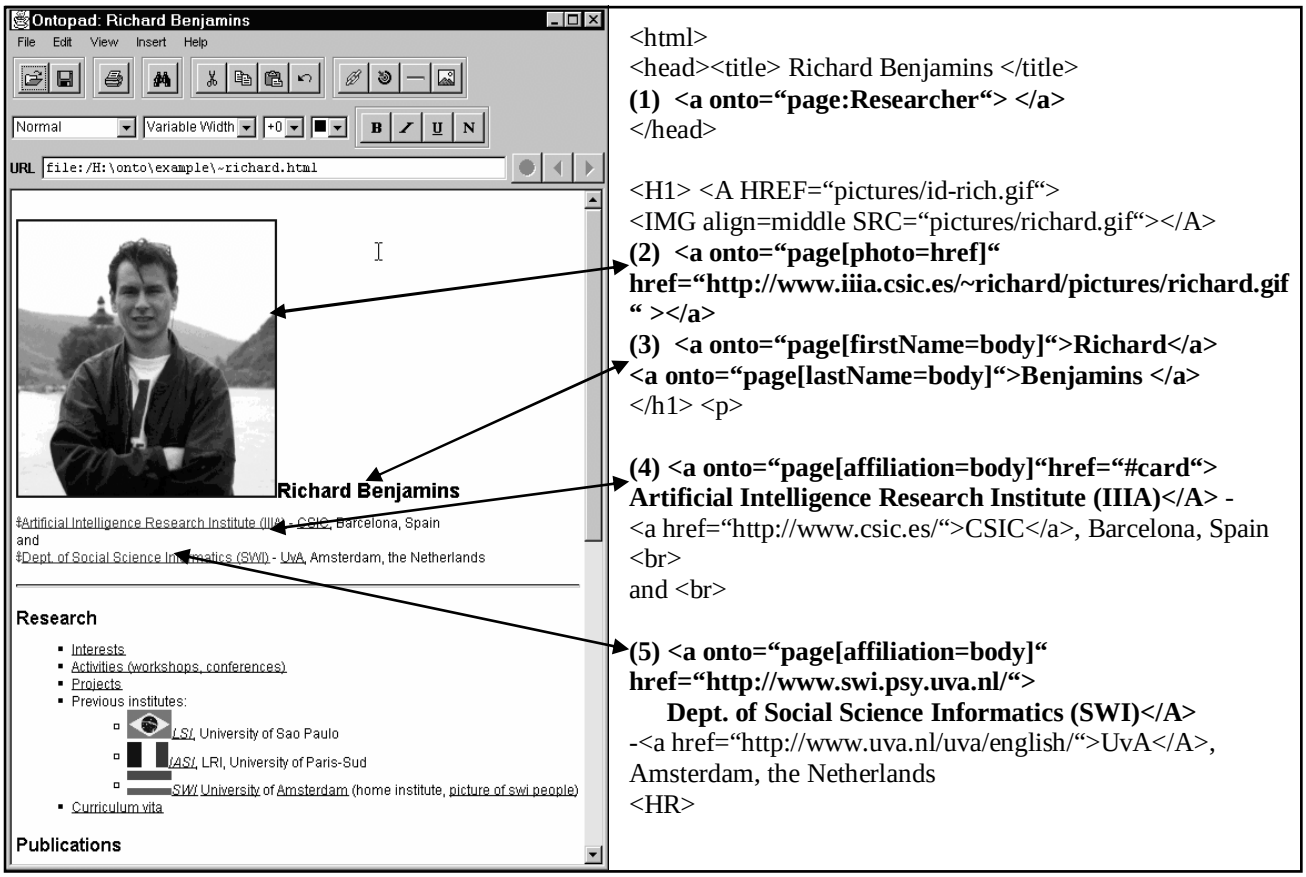

Abbildung 5: Beispiel für eine annotierte Internet-Seite [VOLZ01]

Abbildung 5 ist ein Beispiel für eine annotierte Internet-Seite und zeigt die Homepage von Richard Benjamins sowie einen Ausschnitt aus dem Quelltext der Internet-Seite mit der URL „http://www.iiia.csic.es/richard“.
Das Schlüsselwort “page“ (siehe (1) in Abb.4) repräsentiert die gesamte Internet-Seite, auf der die ontologische Annotationen enthalten sind. Diese Internet-Seite wird als Objekt gesehen, das durch seine URL identifiziert wird und im Beispiel als Instanz der Klasse Forscher (Researcher) festgelegt wird. Richard Benjamins wird also von der URL seiner Homepage als Forscher gekennzeichnet.
Jede Klasse ist mit verschiedenen Attributen verbunden, wobei jede Instanz einer Klasse Werte für diese Attribute definieren kann. Die Klasse Forscher hat nun unter anderem die Attribute Foto (siehe (2) in Abb.4), Vorname, Nachname (siehe (3) in Abb.4) und Zugehörigkeit (siehe (4) und (5) in Abb.4). Diesen Attributen wird jeweils ein Wert zugewiesen. Diese Werte stammen von dem „href“ bzw. „body“- Attribut des „anchor“ Tag. In dem Beispiel sind die Werte für das Foto: „http://www.iiia.csic.es/richard/pictures/richard.gif“, für den Vornamen: „Richard“, für den Nachnamen: „Benjamins“ und für die Zugehörigkeit: „Artificial Intelligence Research Institute“ bzw. „Dept. of Social Science Informatics“ (siehe[DEF+98]).

- **Inference Engine -** Die Inference Engine arbeitet im Hintergrund. Es holt Fakten aus der Datenbank, leitet neues Wissen aus diesen Fakten ab und speichert die Ergebnisse wieder in der Datenbank. Dabei verwendet sie die Schlussfolgerungsregeln der Ontologie und wandelt damit implizites Wissen in explizites Wissen um.
- **Query Engine -** Die Query Engine erhält Anfragen vom Benutzer und beantwortet sie, indem sie den Inhalt der Datenbank durchsucht. Die zugrundeliegende Ontologie dient hier der Unterstützung der Suchanfrage, indem sie z.B. die Begriffe, nach denen gefragt werden kann, definiert. Die Anfrage selbst ist ein Ausdruck in F-Logic.
- **DB-Manager -** Der DB-Manager sorgt für die Verbindung der drei vorherigen Komponenten. Er erhält die Fakten vom Info Agent, tauscht diese mit der Inference Engine und stellt für die Query Engine Fakten zur Verfügung.

4.1.2 Funktionsweise

Die Version von Ontobroker, die von der Firma Ontoprise vertrieben wird, kann auf verschiedene Arten genutzt werden. Einmal als eigenständige Java-Anwendung. Diese liest Dateien mit Fakten, Regeln und Anfragen ein, wertet die Anfragen dann aus und gibt schließlich die Antworten zurück. Der Nachteil hierbei ist, dass die Dateien jedes Mal analysiert und übersetzt werden müssen, was sehr zeitaufwändig ist. Um dieses Problem zu umgehen, kann Ontobroker auch als Server genutzt werden. Dieser Server liest bei seinem Start Dateien mit Fakten und Regeln ein und läuft die ganze Zeit über im Hintergrund. Anfragen werden bei Bedarf an den Server geschickt und von diesem anhand der vorhandenen Fakten und Regeln beantwortet. Außerdem kann Ontobroker in andere Anwendungen integriert werden.
Es gibt bereits eine Reihe von Anwendungen, in die Ontobroker integriert wurde. Zu den bekanntesten gehört die (KA)2 Initiative[8], die den semantischen Zugriff auf alle Dokumente von Forschungsgruppen, die der Wissensgemeinschaft angehören, ermöglicht. Allerdings ist eine Integration von Ontobroker in Internet- bzw. Intranet-Anwendungen nur bei annotierten Seiten möglich. Da bisher nur sehr wenige Internet-Seiten annotiert sind, ist Ontobroker nicht in der Lage, das gesamte Internet nach Informationen zu durchsuchen.
Zwar steigt die Verbreitung von Annotationssprachen, doch ist der Aufwand, alle bereits vorhandenen HTML-Seiten zu annotieren, zu groß. Da die Web Gemeinde sehr groß und heterogen ist, ist es nicht möglich, mit einer einzigen Ontologie alle Inhalte des Internets zu beschreiben. Es wird verschiedene sogenannte Ontogruppen geben, die sich für ihr jeweiliges Thema auf eine Ontologie einigen. Deshalb stellt Ontobroker keinen Ersatz für bestehende Internet-Suchmaschinen wie Google dar, vielmehr wird der Schwerpunkt auf der Integration von Ontobroker in Intranetanwendungen sowie in Internetanwendungen einzelner Gruppen liegen.

4.2 On-To-Knowledge Projekt

Das europäische On-To-Knowledge Projekt (2000–2002) wurde entwickelt, um Methoden und Werkzeuge zu schaffen, die Wissensmanagement im Web und in großen Firmen erleichtern sollen. Die Entwicklungsziele des Projektes waren eine Werkzeug-Umgebung für semantische Informationsverarbeitung und für den Benutzerzugriff, OIL, eine Ontologiesprache (siehe Kapitel 2.2.2) und eine Methodologie zur Einführung von Ontologie-basierten Systemen in Unternehmen sowie deren Validierung in industriellen Fallstudien. Die Partner des Projektes waren: Freie Universität von Amsterdam(NL), coordinator, British Telecom (UK), Swiss Life (CH),

[8] http://ka2portal.aifb.uni-karlsruhe.de/

AIdministrator (NL), CognIT (NO), EnerSearch (SE), AIFB Universität von Karlsruhe (D), OntoText Lab (BU).
Das zentrale Ergebnis dieses Projektes ist eine Ontologie-basierte Werkzeug-Umgebung für Wissensmanagement, wobei der Fokus auf der Unterstützung großer Firmen-Netzwerke und des Internets lagen (siehe [FHD+03]).

4.2.1 Architektur

Die entwickelte Werkzeug-Umgebung soll den Anwendern beim effizienten, natürlichen und intuitiven Zugriff auf firmenweite Informationsquellen helfen und sie ebenfalls bei deren Pflege, Umformung und Gewinnung unterstützen.

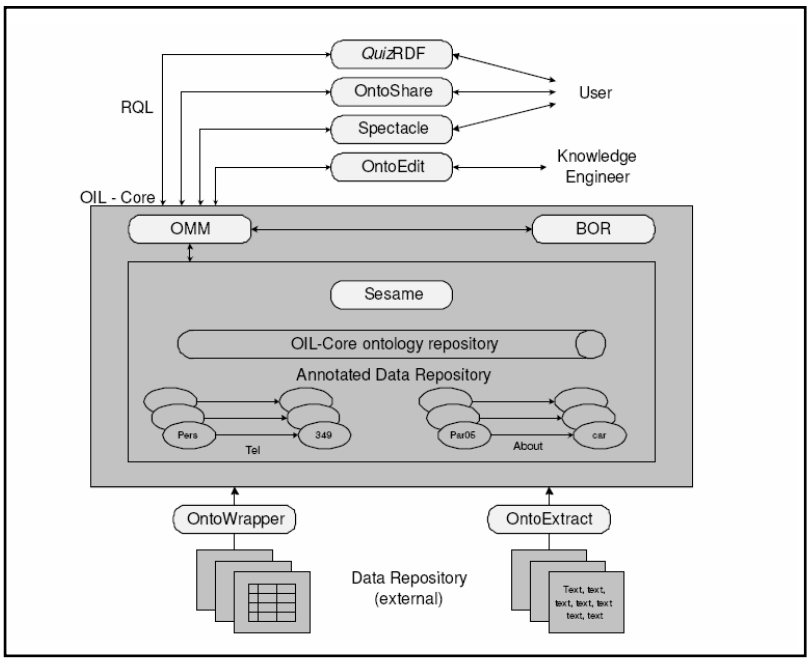

Abbildung 6: On-To-Knowledge Architektur [BERG02]

Abbildung 6 beschreibt den Aufbau des On-To-Knowledge Projekts. Die Komponenten sind in eine dreischichtige Architektur integriert:

- Auf der untersten Ebene extrahieren Werkzeuge wie OntoWrapper und OntoExtract Metadaten aus semistrukturierten und strukturierten Dokumenten und generieren daraus Ontologien.
- Auf der mittleren Ebene werden diese Metadaten von dem Sesame-System in einem Datenrepositorium gespeichert und können abgefragt werden. Das Ontology Middleware Module[9] (OMM) ist eine Erweiterung von Sesame. BOR[10] erweitert die Funktionalität von Sesame um „Reasoning“-Dienste. „Reasoning“ bezeichnet einen Prozess, bei dem aus vorhandenem Wissen neue Information gewonnen wird.
- Auf der obersten Ebene können Kunden und Anbieter Werkzeuge entwickeln, um Wissensdomänen zu untersuchen und zu modifizieren. Mit der RDF-Anfragesprache RQL (RDF Query Language) werden Anfragen auf die vorhandenen Datenquellen gestartet. Aus dem Projekt entstandene Werkzeuge sind OntoEdit, Spectacle, OntoShare und QuizRDF.

9 http://www.ontotext.com/omm
10 http://www.ontotext.com/bor

QuizRDF (siehe [DAWK]) kann wie eine übliche Suchmaschine verwendet werden und berücksichtigt darüber hinaus bei der Seitenindexierung auch RDF-Annotationen.

4.2.2 Sesame

Sesame (siehe [BRÜG03]) ist ein System, welches das persistente Speichern, die Archivierung und Abfrage von RDF und RDFS-Daten ermöglicht. Alle Daten werden bei Sesame in einem Repositorium gespeichert, welches durch ein Datenbanksystem realisiert wird. Sesame ist unabhängig von dem benutzten System und kann neben relationalen Datenbanken auch beispielsweise objektorientierte Datenbank-Management-Systeme für seine Zwecke nutzen. Abbildung 7 zeigt die einzelnen Schichten, die von Sesame verwendet werden.

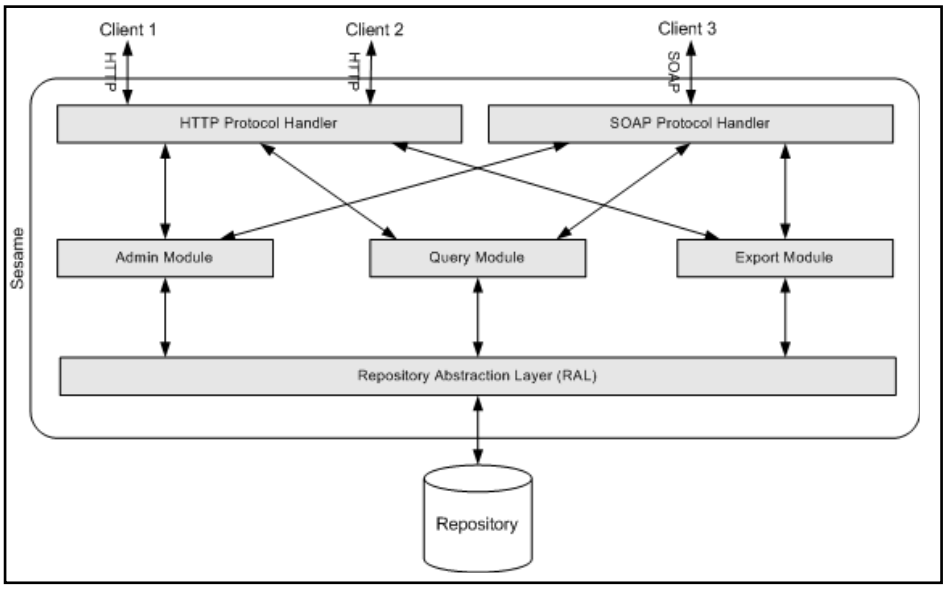

Abbildung 7: Sesame-Architektur

Den Kern der Sesame-Architektur bildet eine Abstraktionsschicht: Repository Abstraction Layer (RAL). Sie stellt RDF-spezifische Methoden zur Verfügung, um auf die RDF-Daten in der Datenbank zuzugreifen. Ein Grund für die Datenbankunabhängigkeit von Sesame ist diese Schicht. Damit ist es möglich, für verschiedenste Datenbanksysteme die Schnittstelle individuell zu implementieren. Auf dem RAL bauen drei Module auf, denen unterschiedliche Aufgabenbereiche zukommen. Das Admin-Modul ist für das Einfügen neuer RDF-Daten und für das Löschen des gesamten Datenbestandes zuständig. Das Query-Modul ist für das Ausführen von Abfragen auf den Datenbestand verantwortlich und das Export-Modul bietet die Möglichkeit, RDF-Daten in XML-Syntax zu exportieren.
Die Kommunikation mit diesen Modulen kann bei Sesame über verschiedene Kommunikationsarten erfolgen, welche von Clients genutzt werden können. Von Sesame wird HTTP (für Zugriffe über das Internet) und SOAP (Simple Object Access Protocol) unterstützt.

4.2.3 Ontology Middleware Module

Das Ontologie Middleware-Modul (OMM) ist eine Erweiterung von Sesame und kann als eine "administrative" Software-Infrastruktur gesehen werden.

Die wichtigsten Komponenten, die unterstützt werden, sind:

- Versionsmanagement für Ontologien,
- Anwenderprofile und Gruppen, um die Rechte für den Zugriff und Modifikationen von Ontologien zu kontrollieren und somit für mehr Sicherheit zu garantieren,

- Unterstützung für Meta-Eigenschaften (wie Status, Last-Updated-By, Verantwortlich, Kommentare, usw.) für ganze Ontologien, wie für separate Konzepte und Eigenschaften. Das Aufsuchen von Ontologien und Eigenschaften gemäß Meta-Informationen ist möglich,
- Zugriff über eine Anzahl an Protokollen: HTTP, RMI, EJB, CORBA und SOAP.

4.2.4 OntoEdit

OntoEdit[11] ist ähnlich wie Protégé-2000 (siehe Kapitel 3) ein Werkzeug zur Erstellung und Wartung von Ontologien. Es wurde ebenso wie Ontobroker von der Firma Ontoprise entwickelt. Abbildung 8 zeigt die graphische Oberfläche von OntoEdit. Man kann eine Hierarchie von Konzepten bearbeiten. Die Konzepte können konkret oder abstrakt sein. Wie bei Protégé-2000 können Plugins installiert werden. Jedoch fehlt bei OntoEdit die Möglichkeit „Forms“ und „Queries“ zu verwenden. Dafür wird aber die Entwicklung von Ontologien mit Hilfe von graphischen Mitteln unterstützt (siehe Abbildung 8).
Wenn die Ontologie mit all ihren Konzepten, Relationen und Axiomen erstellt wurde, kann sie in OntoEdit in verschiedenen Formaten abgespeichert werden. Zum einen im OntoEdit eigenen Format „oxml“, zum anderen aber auch in anderen Formaten wie beispielsweise RDF oder F-Logic.

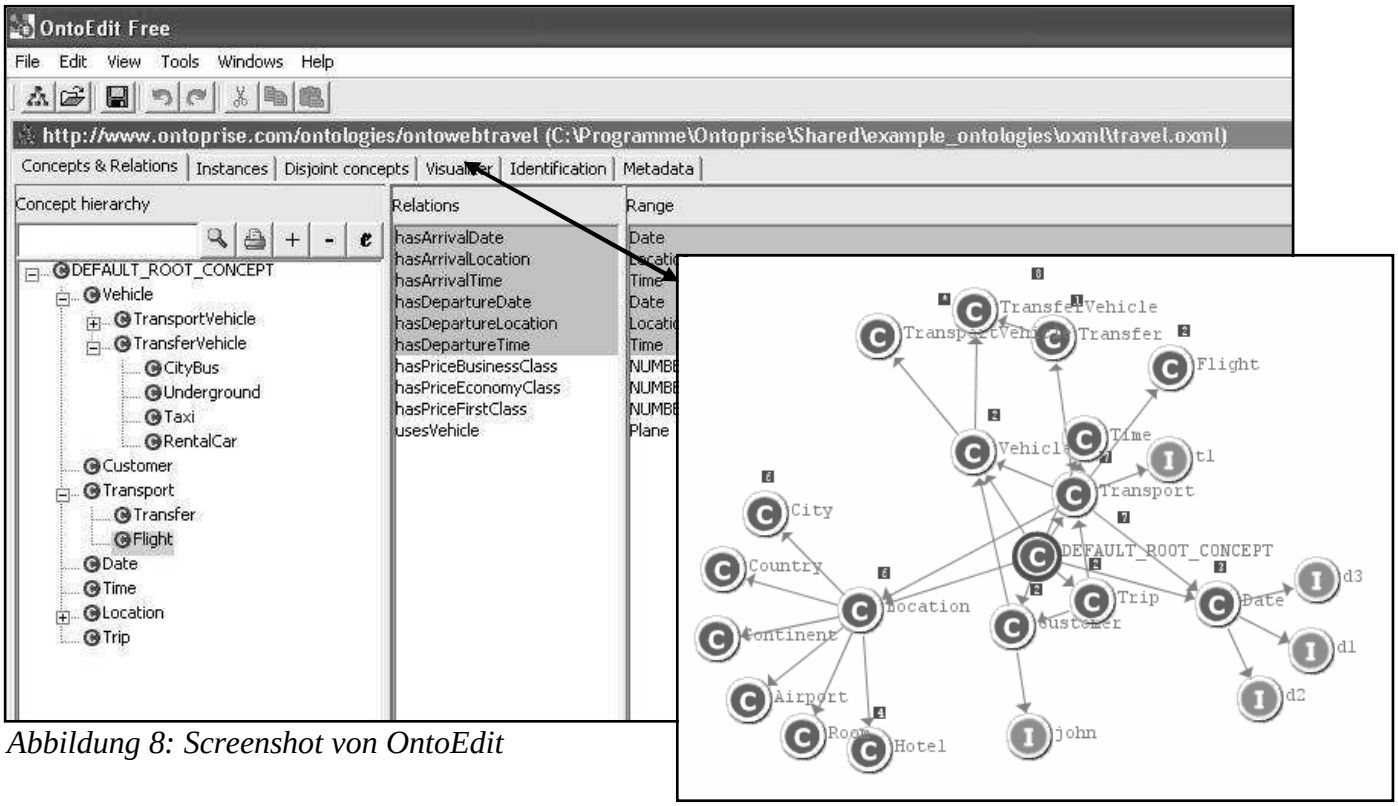

Abbildung 8: Screenshot von OntoEdit

4.2.5 Spectacle

Spectacle [FLHM02] ist eine Plattform, die durch maßgeschneiderte Informationsdarstellung gekennzeichnet ist. Sie soll das Erzeugen jener Informationspräsentationen erleichtern, die für den Anwender interessant sind. Diese Informationen werden dann gegliedert und formatiert dargestellt.
Das Sammeln von Informationen wird einfacher, indem der Benutzer nur noch durch die Informationsrepräsentation navigieren muß und mit wenigen Mausklicken die gewünschte Information findet. Dadurch gestaltet sich das Suchen effizienter, die Genauigkeit höher und der Zeitaufwand geringer.

[11] http://www.ontoprise.de/products/ontoedit

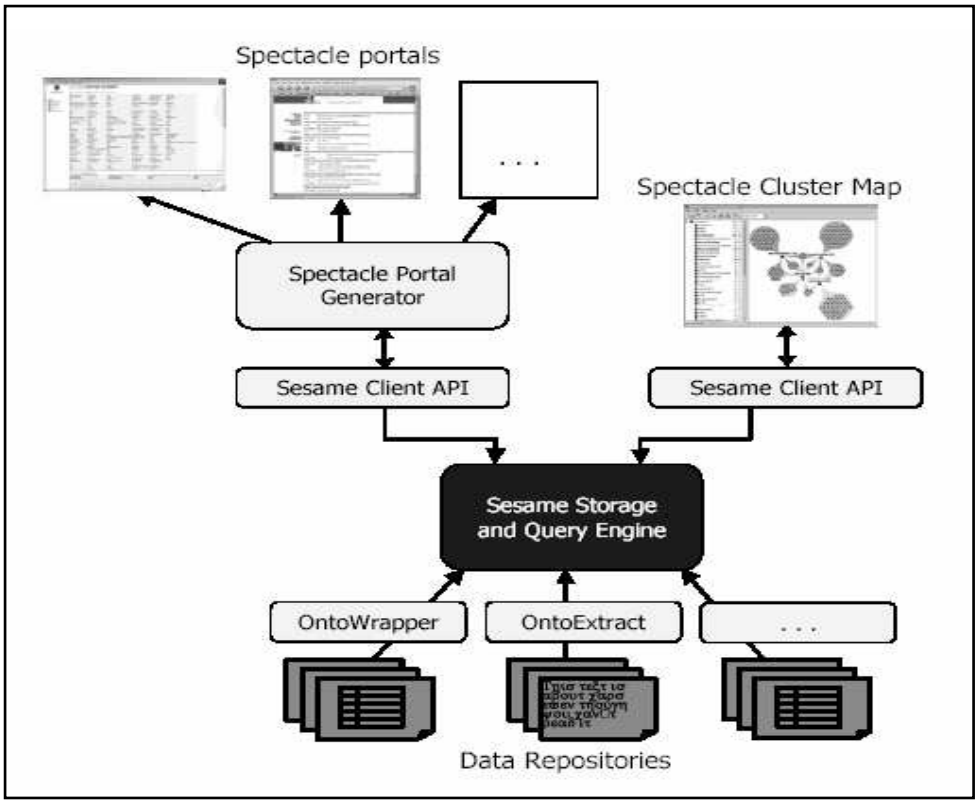

Abbildung 9: Architektur von Spectacle

Abbildung 9 beschreibt die Architektur von Spectacle. Zwei Spectacle Komponenten, der Spectacle Portal Generator und der Spectacle Cluster Map Viewer, greifen über die Sesame Client API auf Informationen aus dem Speicher zu. Der Portal Generator holt sich Informationen, um Portale für verschiedene Aufgaben und Anwendertypen zu erstellen und der Cluster Map Viewer lädt die RDFS Klassenhierarchie und Instanzen über die Sesame Client API. Der Benutzer kann dann graphisch durch diese Taxonomie navigieren.
Im Folgenden werden die zwei Visualisierungsmöglichkeiten von Spectacle beschrieben.

- **RDF Explorer -** Der RDF Explorer ist ein Portal, das auf einer Sesame-Quelle aufbaut. Er soll Vorführungszwecken dienen und kann ebenfalls als Ausgangspunkt für die Erstellung anderer Portale, die auf spezielle Domänen oder Aufgaben abgestimmt sind, dienen. Auch beim Untersuchen von RDF Quellen ist er hilfreich, besonders dann, wenn die Definitionen von Schemata unbekannt sind.
  Abbildung 10 stellt einen Screenshot des RDF Explorers dar, der eine ausgewählte Museumsklassifizierung zeigt.

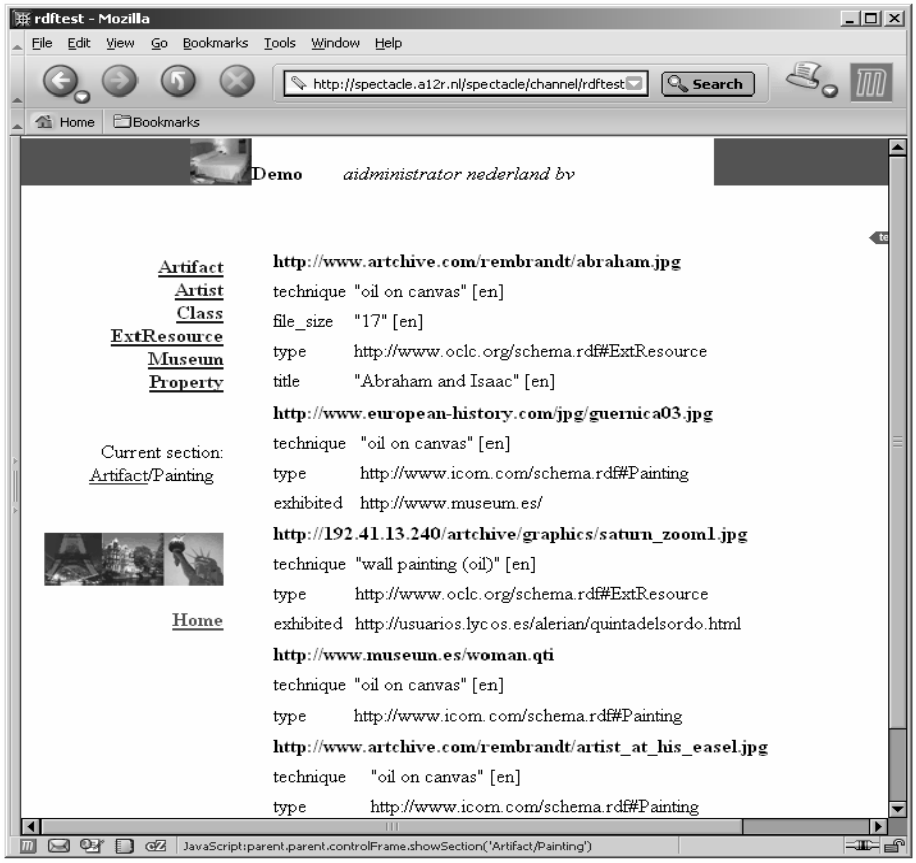

Abbildung 10: Screenshot von RDF Explorer

Die Klassifizierung enthält Eigenschaften wie Künstler, Kunstgegenstände, Museen und die Beziehungen untereinander. Die Klassen in dieser Hierarchie werden in eine hierarchisch geordnete Sammlung von Seiten übersetzt. (Der Screenshot zeigt die Malerei-Klasse, welche eine Unterklasse der Kunstgegenstände-Klasse ist.) Jede Seite fasst die Kunstwerke zusammen, die Instanzen von der repräsentierten Klasse sind. Jedes Kunstwerk wird durch seine URI und Eigenschaften des Kunstwerkes dargestellt.
Das Portal hat jedoch keine Kenntnisse über die dargestellte Domäne, welche bei der Darstellung berücksichtigt werden könnten. So wird zum Beispiel nicht erkannt, dass die URI eines Gemäldes ein Bild im Web darstellt, welches in die Seite eingefügt werden könnte. Dies sind große Einschränkungen in der Brauchbarkeit und Nutzbarkeit des Portals. Zusätzliche Erweiterungen sind definitiv notwendig, damit ein brauchbares Portal geschaffen wird.

- **Spectacle Cluster Map Viewer -** Der Spectacle-Cluster-Map Viewer ist eine Anwendung, die der Visualisierung von instanzierten Taxonomien wie Klassenhierarchien dient. Er zeigt die Instanzen von Klassen an, die vom Benutzer interaktiv ausgewählt werden. Der Spectacle Cluster Map Viewer verdeutlicht welche Instanzen zu mehreren Klassen gehören, indem er die Verbindungen zu jeder Klasse kenntlich macht. Durch diese Darstellung versteht der Benutzer, aufgrund der von den Klassen geteilten Instanzen, wie die Klassen miteinander verwandt sind. Dadurch wird die Analyse von hierarchisch geordneten Daten erleichtert.

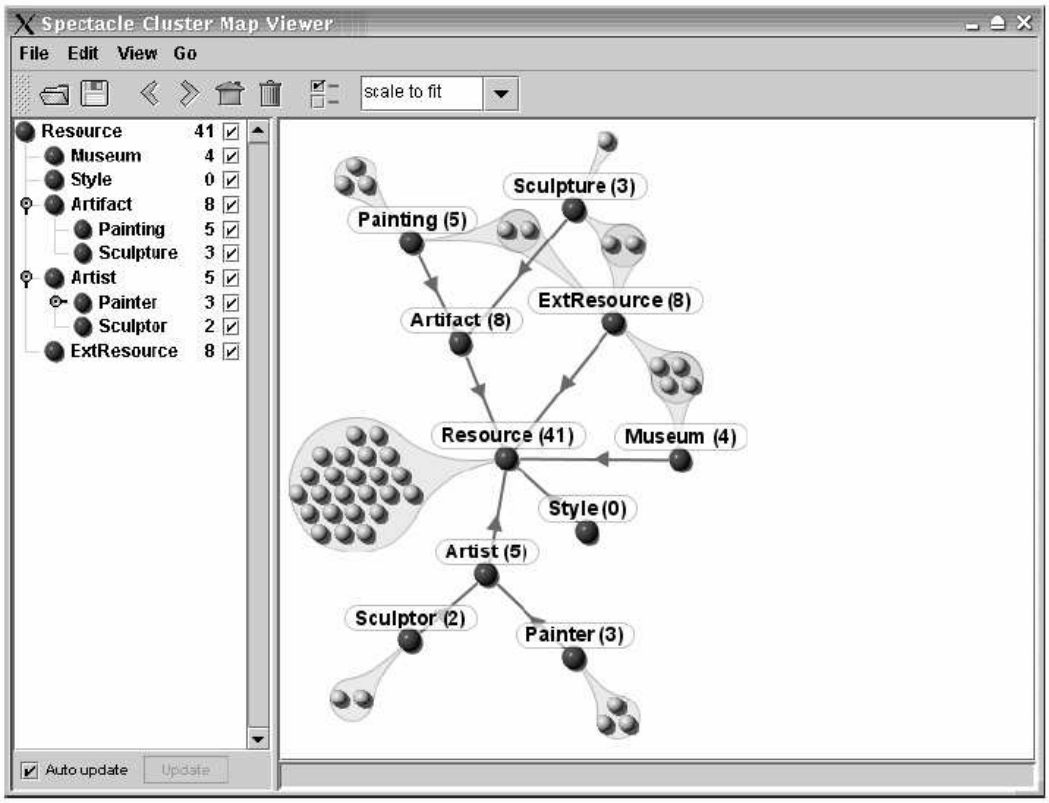

Abbildung 11: Screenshot vom Spectacle Cluster Map Viewer

Das Cluster Map Viewer-Beispiel zeigt in Abbildung 11 den Datenbestand eines Museums. Alle Klassen werden durch grüne Kreise dargestellt. Die kleinen gelben Kugeln sind Instanzen der Klassen. Jede Instanz ist durch die grünen Verbindungen mit ein oder mehreren Klassen verbunden, wodurch die Zugehörigkeit ausgedrückt wird.

4.2.6 OntoShare

OntoShare [DADS] ist eine Ontologie-basierte Wissensumgebung, auf die eine Arbeitsgruppe gemeinsam zugreifen kann. Dabei werden die Interessen eines jeden Anwenders in Form eines Anwenderprofils dargestellt. Mit OntoShare können Internet-Seiten und Informationsquellen von anderen Anwendern zusammengefasst und Schlüsselwörter extrahiert werden. Diese Informationen werden dann wieder mit anderen Anwendern geteilt, die ebenfalls Interesse an der jeweiligen Information bekundet haben. OntoShare wird also verwendet, um Informationen zu speichern, wiederzugewinnen und zusammenzufassen und auch um andere Anwender über vorhandene Informationen zu informieren, die für sie in gewissem Sinne wertvoll sein können.
Jede dieser gemeinsamen Informationen führt zu einem Eintrag im OntoShare Speicher, welcher Ontologien enthält, die in RDF und RDFS repräsentiert werden.

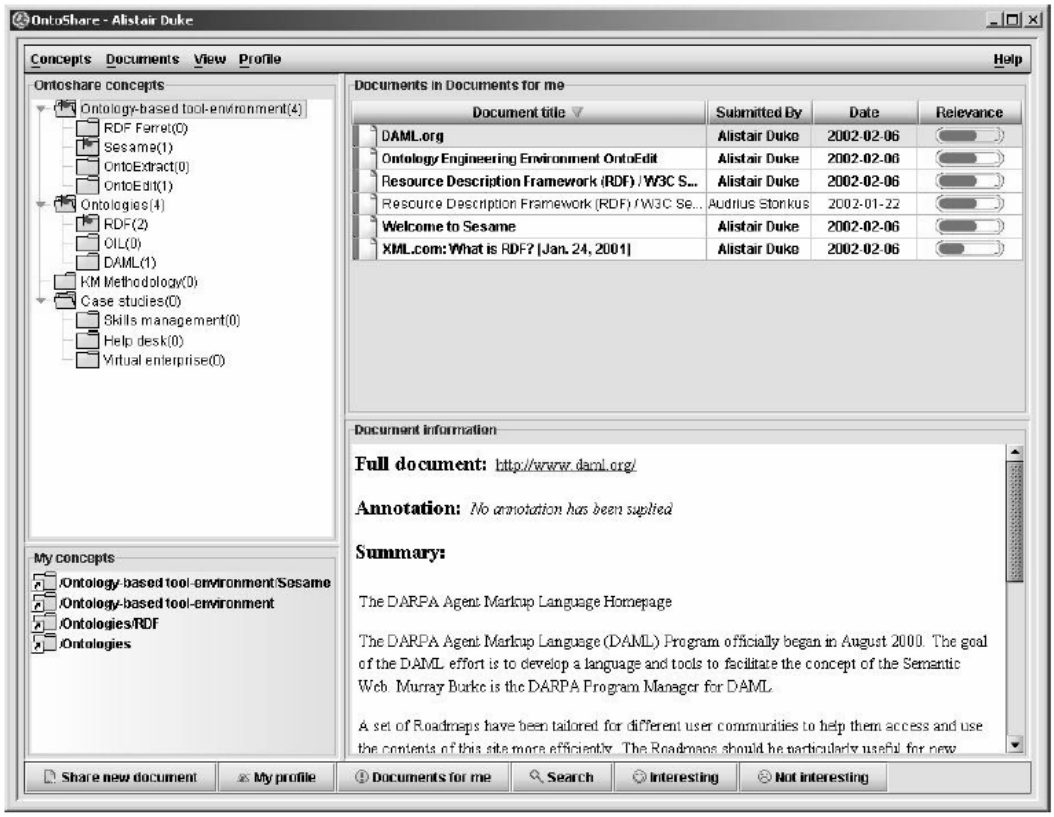

Abbildung 12: Screenshot von Ontoshare

Abbildung 12 zeigt die graphische Oberfläche von OntoShare. Im Folgenden werden Wege dargestellt, wie OntoShare den Zugriff und das automatische Teilen von gemeinsamem Wissen ermöglicht.

- **E-Mail Benachrichtigung** - Wenn Informationen in OntoShare geteilt werden, prüft das System die Profile anderer Anwender aus derselben Arbeitsgruppe. Bei hoher Übereinstimmung der Anwender-Interessen mit den Informationen, wird das Mitglied durch eine automatisch generierte E-Mail benachrichtigt.
- **Suchen im gemeinsamen Speicher** - Analog zu normalen Suchmaschinen kann der Anwender mit Schlüsselwörtern nach Dokumenten im gemeinsamen Speicher suchen, die die gewünschten Informationen enthalten.
- **Personalisierte Informationen** - Im Feld „Documents for me“ in Abbildung 12 kann der Anwender die Dokumente einsehen, die am besten zu seinem Profil passen. Unten kann der Anwender mit den Buttons „Interesting“ oder „Not Interesting“ bestimmen, ob er an ausgewählten Dokumenten interessiert ist. In diesem Fall kann sich die Entscheidung nach Wunsch des Anwenders auf sein Profil auswirken.

4.2.7 Die On-To-Knowledge Methologie

Im Rahmen des On-To-Knowledge Projekts wurde eine Methodologie entwickelt, um eine Ontologie im Rahmen einer Ontologie-basierten Anwendung mit Hilfe von On-To-Knowledge Werkzeugen zu entwickeln.

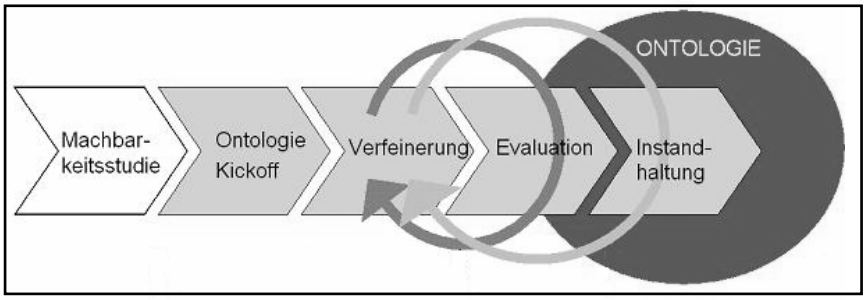

Abbildung 13: Vorgehensweise zur Ontologieerstellung

Wie in Abbildung 13 zu erkennen, besteht der Entwicklungsweg einer Ontologie aus fünf Hauptphasen:

- **Machbarkeitsstudie -** In der Machbarkeitsstudie werden alle Personen identifiziert, die an dem Projekt teilnehmen, der Bereich für die ontologiebasierte Anwendung fokusiert und Entscheidungen über die Durchführung von Projekten getroffen. Zudem werden Werkzeuge aus der On-To-Knowledge Umgebung ausgewählt, die den Entwicklungsprozess unterstützen sollen.
- **Ontologie Kickoff -** Das Ergebnis der Kickoff-Phase ist eine Ontologie-Anforderungs-Spezifikation, die eine Beschreibung des Ziels und Anwendungsbereichs der Ontologie sowie des geplanten Einsatzzweckes der darauf aufbauenden ontologiebasierten Anwendung enthält. Die Spezifikation soll den Ontologie-Entwickler beim Einfügen, Entfernen und der hierarchischen Strukturierung von Begriffen in der Ontologie unterstützen. Bereits in dieser frühen Phase der Entwicklung sollte ein Augenmerk auf bereits entwickelte und potentiell wiederverwendbare Ontologien gerichtet werden
- **Verfeinerung** - Ziel der Verfeinerungsphase ist die Entwicklung einer anwendungsbezogenen Ziel-Ontologie gemäß der Spezifikation aus der Kickoff- Phase.
- **Evaluation -** Die Evaluationsphase dient zur Bewertung der Nützlichkeit der entwickelten Ontologie und der darauf aufbauenden Softwareumgebung. In einem ersten Schritt testet der Ontologie-Entwickler, ob die Ziel-Ontologie den Anforderungen der Ontologie-Anforderungs-Spezifikation genügt. In einem zweiten Schritt wird die Ontologie in ihrer Anwendungsumgebung getestet. Feedback von Beta-Testern kann wertvolle Hinweise für die Entwicklung der Ontologie liefern und weitere Verfeinerungsschritte initiieren. Diese Phase ist eng mit der Verfeinerungsphase verbunden und ein Ontologie-Entwickler muss eventuell mehrere Zyklen durchlaufen, bis die Ziel-Ontologie den gewünschten Detaillierungsgrad erreicht hat. Das Überführen der Ontologie in die praktische Anwendung beendet die Evaluationsphase.
- **Instandhaltung -** Die “echte Welt” ändert sich ständig, wie auch die Anforderungen an Ontologien. Um diese Änderungen in der Modellierung der Ontologie widerzuspiegeln, muss sie in regelmässigen Abständen instandgehalten werden. Empfehlenswert ist ein verantwortlicher Ontologie-Entwickler, der die geänderten Anforderungen sammelt, ausführt und neue Versionsstände der Ontologie freigibt. Die Freigabe sollte erst nach Tests erfolgen, um mögliche Auswirkungen in der Anwendung nachzuvollziehen. Dies erfordert unter Umständen mehrere Verfeinerungszyklen. Ontologien müssen während der gesamten Lebenszeit instand gehalten werden.

5 Methoden zur Extraktion und Verwendung von taxonomischen Relationen

Wie anhand der Ontologie-basierten Systeme in Kapitel 4 dargestellt, werden zur Gewinnung von Daten Informationen aus Texten und Dokumenten extrahiert. Aus den Informationen werden dann neue Ontologien generiert oder sie werden in schon vorhandene taxonomische Strukturen eingeordnet.
In diesem Kapitel werden verschiedene Techniken zum Erstellen und Verwenden solcher taxonomischen Strukturen vorgestellt, um den Ontologie-Erzeugungsprozess, der sehr zeitaufwendig sein kann, zu automatisieren (siehe [MAPS03]).

5.1 Syntaktische Analyse

Die Idee formal-grammatische Muster in Form von regulären Ausdrücken zu gebrauchen, um semantische Relationen, insbesondere taxonomische Relationen, zu extrahieren, wurde von Hearst (siehe [HEAR92]) eingeführt. Dabei wird der Text nach Instanzen von bekannten lexikalisch-syntaktischen Mustern durchsucht, aus denen man dann taxonomische Relationen erhält.
Beispiel: Folgendes formal-grammatisches Muster wird betrachtet (NP = Nominalphrase steht hier für ein beliebiges Substantiv):
…NP {,NP} * {,} oder andere NP …
Wenn man dieses Muster auf einen Satz anwendet, kann man folgern, das die NPs links von "oder andere" Subkonzepte von dem NP rechts von "oder andere" sind.
Aus dem Satz
„Quetschungen, Wunden, gebrochene Knochen oder andere Verletzungen kommen häufig vor."
kann man die taxonomischen Relationen
(Quetschung, Verletzung),
(Wunde, Verletzung),
(gebrochener Knochen, Verletzung) extrahieren.

Bei dieser Methode werden die Muster manuell definiert, was sehr zeitaufwändig und fehleranfällig ist. Deswegen wurde dieser Ansatz durch ein symbolisch, maschinell lernendes Werkzeug erweitert, um regelhaftes Wissen aus Daten zu gewinnen und damit lexikalische Muster zu verfeinern. In diesem Zusammenhang wurde das PROMETHEE[12] System präsentiert; es unterstützt den halb automatischen Erwerb von semantischen Relationen und die Verfeinerung von formal grammatischen Mustern.

5.2 Statistisch basierte Analyse

In diesem Abschnitt wird eine Methode zum Erstellen von taxonomischen Beziehungen betrachtet, die die Semantik einer neuen Klasse folgern und diese in Beziehung zu vorhandenen Klassen aus der Ontologie setzen. Dies geschieht auf der Basis von statistischen Daten.
Die Idee hinter dieser Methode ist die Tatsache, dass die semantische Identität eines Wortes anhand seiner Verteilung über verschiedene Kontexte wiedergegeben wird, so dass die Bedeutung eines Wortes durch das gleichzeitige Auftreten anderer Wörter (distributionale Daten) und der Häufigkeiten dieses Auftretens repräsentiert wird. Dadurch müssen keine besonderen Vorkehrungen mehr getroffen werden, wie formal-grammatische Muster, sondern man steuert darauf zu, den Prozess des taxonomischen Lernens zu automatisieren.
Um nun existierende Ontologien mit Hilfe von statistischen Daten über Klassen zu erweitern, werden automatische Klassifikationsmethoden angewandt, um einen passenden Platz für die neue

[12] http://www.greyc.ismra.fr/~regis/Promethee/

Klasse in der Ontologie zu ermitteln. Im Folgenden wird mit dem knn-Verfahren eine solche Methode vorgestellt.

Verfahren der k-nächsten Nachbarn (knn-Verfahren)
Mit dem knn-Verfahren wird die Zugehörigkeit eines Objektes zu einer Klasse von mehreren möglichen Klassen bestimmt. Die möglichen Klassen sind durch ihre Objekte bestimmt. Zum unbekannten Objekt werden k –das ist eine festgelegte Anzahl– Objekte ermittelt, die mit ihm die größte Ähnlichkeit haben. Das zu klassifizierende Objekt wird der Klasse zugeordnet, der die meisten der "k nächsten Nachbarn" angehören.
Wird nun eine neue Klasse in eine Ontologie eingeführt, so wird sie das Hyponym (Unterklasse) jener Klasse, dessen Anzahl von Hyponymen mit nächsten Nachbarn am größten ist.
Ein Nachteil des knn-Verfahren sind die hohen Kosten, die bei den Ähnlichkeitsberechnungen entstehen, denn das neue Objekt muss mit allen Objekten verglichen werden.

5.3 Verwendung der taxonomischen Struktur einer Ontologie

Hier werden die Möglichkeiten untersucht die Informationen über die taxonomische Organisation einer Ontologie, mit den statistischen Daten über die Konzepte zu kombinieren, um damit neue Objekte in Klassen einzufügen. Folgende zwei Algorithmen realisieren diesen Ansatz.

5.3.1 Tree Descending Algorithmus

Diesem Ansatz liegt zugrunde, dass die Semantik jeder Klasse jeweils nur die wichtigsten semantischen Charakteristika untergeordneter Klassen widerspiegeln. Um ein neues Wort in eine Klasse des Ontologiebaumes einzuordnen, steigt man von der Wurzel bis zu den Blättern hinab. Bei jedem Knoten wird entschieden, welchem Pfad zu folgen ist, indem man die Kind-Klasse, mit der das neue Wort die meisten Gemeinsamkeiten hat, wählt. Nachdem man mit der Suche ein Blatt erreicht hat, wird das Wort der Klasse auf dem Pfad zugeordnet, dessen Objekte die größte Ähnlichkeit mit dem Wort haben.

5.3.2 Tree Ascending Algorithmus

Eine weitere Möglichkeit, Informationen über die Beziehungen zwischen Klassen in einer Ontologie, bei der Entscheidung, ein neues Wort in eine Klasse einzuorden, zu nutzen, ist es die Messungen von taxonomischer Ähnlichkeit und distributionaler Ähnlichkeit zwischen nächsten Nachbarn zu kombinieren.
Angenommen Wörter, die für ein gegebenes neues, einzuordnendes Wort alle zu den nächsten Nachbarn gehören, befinden sich in verschiedenen Klassen.
Beispielszenario: TRAILER soll in eine neue Klasse eingefügt werden:
Ähnlichkeiten (siehe Abbildung 14) sind nach der knn-Methode zu box (0,9), zu house (0,7), zu barn (0,6) und zu villa (0,5).

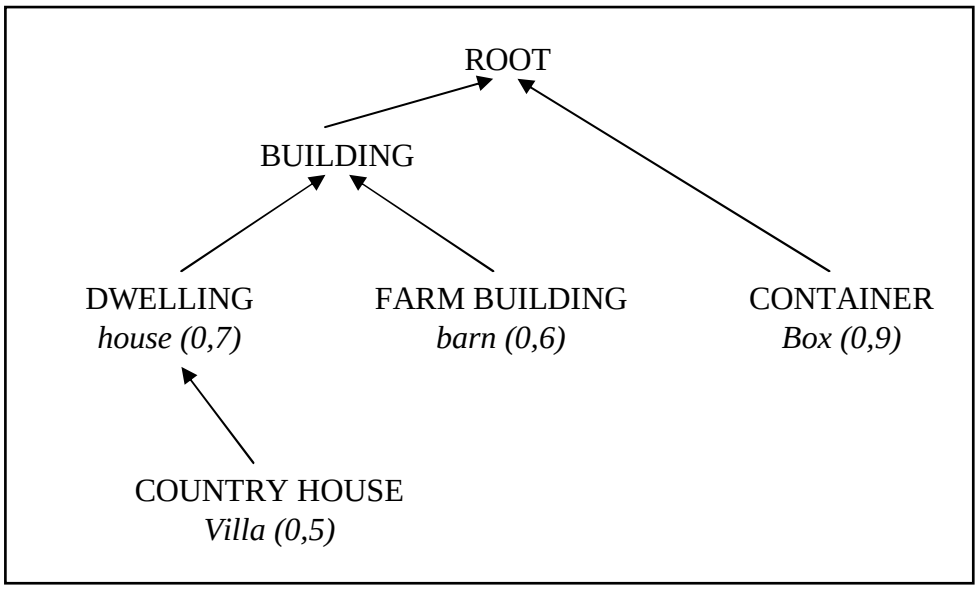

Abbildung 14: Taxonomische Organisation

In diesem Fall würde die knn-Verfahren das Wort trailer in die Klasse CONTAINER hinzufügen, da es zu box die größte Ähnlichkeit zu haben scheint.
Aber unter den nächsten Nachbarn gibt es auch drei Wörter, die sich semantisch ähneln, obwohl sie nicht zur selben Klasse gehören. Deswegen wäre es sicherer, das neue Wort einer Klasse hinzuzufügen, die ein oder alle der drei semantisch ähnlichen Nachbarn einordnet. Hier kämen beipielsweise die Klassen BUILDING oder DWELLING in Frage.
Nun stellt sich die Frage, ob man eine dieser beiden Klassen nehmen soll oder doch die Klasse CONTAINER. Aber wie genau sollen diese Möglichkeiten gegeneinander abgewichtet werden?
Eine Gewichtung für die beiden Klassen BUILDING und DWELLING kann wie folgt vorgenommen werden:
Die taxonomische Ähnlichkeit zwischen zwei Klassen wird wie folgt berechnet. Eine Taxonomie bestehe aus einem Baum mit einer Knotenmenge N und dazugehöriger Kantenmenge $E \subset N \times N$ und einer gemeinsamen ROOT $\in N$. Die erste gemeinsame Oberklasse (least common superconcept, lcs) von einem Klassenpaar a, b ist folgendermaßen definiert:

$lcs(a,b) := c \in N : \delta(a,c) + \delta(b,c) + \delta(Wurzel,c)$ mit c minimal
Dabei beschreibt $\delta(a,b)$ die Anzahl der Kanten des kürzesten Weg zwischen a und b.

Die taxonomische Ähnlichkeit zwischen a und b ist gegeben durch

$$\Omega(a,b) := \frac{\delta(Root,c)}{\delta(a,c) + \delta(b,c) + \delta(Root,c)} \quad \text{mit } c = lcs(a,b).$$

Ω hat also immer einen Wert zwischen 0 und 1, wobei 1 die maximale Ähnlichkeit beschreibt.
Die gewichtete Ähnlichkeit W für ein Kandidatkonzept n wird berechnet, indem die distributionalen Ähnlichkeiten sim(t,h) des einzuordnenden Wortes t zu den Hyponymen h des Kandidatenkonzeptes aufsummiert werden, wobei jede Ähnlichkeit mit der taxonomischen Ähnlichkeit $\Omega(n,h)$ zwischen dem Hyponym h und dem Kandidatenkonzept n aufgewichtet wird:

$$W(n) := \sum_{h \in I_n} sim(t,h) \cdot \Omega(n,h)$$

I_n sind die Hyponyme unterhalb des Kandidatenkonzeptes n.
Im obigen Beispiel ergeben sich für die beiden Kandidatenklassen BUILDING und DWELLING folgende Werte:
n = BUILDING (lcs c = BUILDING):

$$W(n) = 0{,}7 \cdot \frac{1}{1+0+1} + 0{,}6 \cdot \frac{1}{1+0+1} + 0{,}5 \cdot \frac{1}{2+0+1} = 0{,}7 \cdot 0{,}5 + 0{,}6 \cdot 0{,}5 + 0{,}5 \cdot 0{,}\overline{3} = 0{,}82$$

n = DWELLING (lsc c = DWELLING):

$$W(n) = 0{,}5 \cdot \frac{2}{1+0+2} = 0{,}5 \cdot 0{,}\overline{6} = 0{,}\overline{3}$$

n = CONTAINER (lsc c = CONTAINER):

$$W(n) = 0{,}9 \cdot \frac{1}{0+0+1} = 0{,}9$$

Nach dieser Methode wird Trailer also in die Klasse Container eingeordnet.

6 Zusammenfassung

Gegenstand dieser Ausarbeitung war die Betrachtung des Themas Web Wissensmanagement, das sich mit dem gezielten Umgang der Wissensbasis des Internets und Firmen-Intranets befasst. In diesem Zusammenhang wurde eine Einführung in das Themengebiet der Ontologien gegeben. Ontologien unterstützen die einheitliche Repräsentation von Wissen sowie das Ableiten von neuem Wissen. Weiterhin wurden die Ontologiesprachen RDF mit RDFS, OIL und F-Logic eingeführt, mit denen Ontologien formal definiert werden können. Ontologiewerkzeuge bieten mit ihren unterschiedlichen Funktionalitäten bei der Entwicklung, Verwendung und Pflege von Ontologien eine entscheidende Hilfe. Anhand von Protégé-2000 wurde ein Ontologie-Erstellungswerkzeug präsentiert. Des Weiteren existieren verschiedenartige Systeme, die auf Ontologien basieren. Sie extrahieren und verarbeiten Daten aus vorhandenen Dokumenten, speichern diese und leiten neues Wissen daraus ab und erlauben es, Anfragen auf diese Daten zu generieren. Diese Systeme sollen das Wissensmanagement unterstützen. In dieser Ausarbeitung wurden mit Ontobroker und dem On-To-Knowledge Projekt zwei Ontologie-basierte Systeme vorgestellt. Schließlich wurden im Bezug auf die Extraktion und die Verwendung von taxonomischen Relationen mehrere Methoden dargeboten.

Abschließend bleibt zu bemerken, dass die Entwicklungen und die Forschung rund um das Thema Web Wissensmanagement noch am Anfang stehen und weiter ausbaufähig sind. Das angestrebte Ziel wird es sein, dem Menschen den Umgang mit Wissen auf der Basis von Ontologie-basierten Systemen so weit wie möglich zu erleichtern.

7 Referenzen

[BERG02] Sonia Bergamaschi;
Ontology dynamics for Semantic Web: the MOMIS approach
Universita Degli Studi Di Modena E Reggio Emilla; 2002
(Elektronisch verfügbar unter: sparc20.dsi.unimo.it/tesi/fergnani.pdf)

[BRÜG03] Daniel Brügge;
Konzeption und Implementierung eines verteilten Agentensystems zur persistenten Speicherung und zum Austausch von RDF-Daten
Bachelorarbeit, Technische Universität München, 15. Juli 2003
(Elektronisch verfügbar unter: www11.informatik.tu-muenchen.de/publications/pdf/ba-bruegged2003.pdf)

[CHAO] Kaies Chaouch;
Ontologien
(Elektronisch verfügbar unter: www.dfki.uni-kl.de/~chaouch/download/pc_4108/chaouch/ DA_K2_Ontologien.doc)

[DAWK] John Davies; Richard Weeks; Uwe Krohn;
QuizRDF: Search Technology for the Semantic Web
(Elektronisch verfügbar unter: www.cs.rutgers.edu/~shklar/www11/final_submissions/paper6.pdf)

[DADS] John Davies; Alistair Duke; Audrius Stonkus;
OntoShare: Using Ontologies for Knowledge Sharing
BTexact Technologies, Orion 5/12, Adastral Park, Ipswich IP5 3RE, UK
(Elektronisch verfügbar unter: http://semanticweb2002.aifb.uni-karlsruhe.de/proceedings/Research/davies.pdf)

[DEF+98] Decker Stefan; Erdmann Michael, Fensel Dieter, Studer Rudi;
ONTOBROKER: Ontology based Access to Distributed and Semi-Structured Information
Tutorial at the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW); Sigüenza, Spain, 01.–04.10.2002, page 26-45

[FHD+03] Dieter Fensel; Frank van Harmelen; Ying Ding, Michel Klein; Hans Akkermans; Jeen Broekstra; Arjohn Kampman; Jos van der Meer; York Sure; Rudi Studer; Uwe Krohn; John Davies; Robert Engels; Victor Iosif; Atanas Kiryakov; Thorsten Lau; Ulrich Reimer; Ian Horrocks;
On-To-Knowledge: Semantic Web Enabled Knowledge Management
Web Intelligence, Springer, Berlin, März 2003, s. 277-299

[FLHM02] Christiaan Fluit; Herko ter Horst; Jos van der Meer;
On-To-Knowledge: Content-driven Knowledge management Tools through Evolving Ontologies
Responsible Partner AIdministrator, 12-9-2002
(Elektronisch verfügbar unter: http://www.ontoknowledge.org/downl/del13.pdf)

[GRÄB03] Dietmar Gräbner;
Web Service Descriptions
Seminar aus Data und Knowledge Engineering, SS2003
(Elektronisch verfügbar unter: http://www.isys.uni.klu.ac.at/ISYS/Courses/03SS/S-DKE/graebner.pdf)

[HEAR92] M. A. Hearst;
Automatic acquisition of hyponyms from large text corpora;
In:Proc. The 14th International Conference on Computational Linguistics,
Nantes, France, 1992

[LESE03] Heiko Leseberg;
Ontologien in Multiagentensystemen Standards und Werkzeuge
Baccalaureatsarbeit zum Projekt „Realisierung verteilter Agentensysteme“,
Fachbereich Informatik, Universität Hamburg, September 2003
(Elektronisch verfügbar unter: vsis-www.informatik.uni-hamburg.de/publications/readstu.phtml/SA/248/leseberg_bacc_03.pdf)

[MAPS03] Alexander Maedche; Viktor Pekar; Steffen Staab;
Ontology Learning Part One – on Discovering Taxonomic Relations form the Web
Web Intelligence, Springer, Berlin, März 2003, s. 301-318

[MMS+03] Alexander Maedche; Boris Motik; Ljiljana Stojanovic; Rudi Studer; Raphael Volz;
Ontologies for Enterprise Knowledge Management
In: IEEE Intelligent Systems, Volume 18, Number 2, March/April 2003, s. 26-33
(Elektronisch verfügbar unter: http://kaon.semanticweb.org/papers)

[ONTO] ontoprise GmbH;
F-Logic Tutorial
(Elektronisch verfügbar unter: http://www.ontoprise.de/documents/tutorial_flogic.pdf)

[SCHW02] Andrea Schwidrowski;
OntoBroker
Seminar Fallbasiertes Schließen, Universität Hildesheim, SS 2002
(Elektronisch verfügbar unter:
http://www.hausarbeiten.de/faecher/download/ing/22317.html)

[STOS01] Rudi Studer; Henrik Oppermann; Hans-Peter Schnurr;
Die Bedeutung von Ontologien für das Wissensmanagement
Institut AIFB, Universität Karlsruhe, September 2001
(Elektronisch verfügbar unter: http://www.ontoprise.de/documents/Bedeutung_von_Ontologien_fuer_WM.pdf)

[VOLZ01] Raphael Volz;
Einführung in Ontologien
Beitrag zum Workshop Begriffliche Formalisierung von Prozessen und Systemen,
TU Dresden, November 2001
(Elektronisch verfügbar unter: www.math.tu-dresden.de/~rudolph/Dresden_Workshop.pdf)