Website-Ontologien für semantische Content Management Systeme

Holger Stitz

DIPLOMARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im September 2011

Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Hagenberg, am 22. September 2011

Holger Stitz

Inhaltsverzeichnis

Kurzfassung

Website-Ontologien stellen eine besondere Form der Ontologien dar, da sie thematisch eng mit der Website verknüpft sind und von dieser veröffentlicht werden. Die Ontologien strukturieren Teile des Seiteninhaltes als Ressourcen und legen das Wissen mit den standardisierten Sprachen des Semantic Webs ab, so dass sie von Mensch und Maschine gleichermaßen verarbeitet werden können.

In dieser Diplomarbeit wird der Erstellungsprozess einer Website-Ontologie in einer teilautomatisierten Lösung für ein Content Management System (CMS) beschrieben. Dazu werden die theoretischen Grundlagen erläutert und der Markt für semantische CMS analysiert. Anschließend wird ein implementiertes Modul für das CMS *Drupal* vorgestellt, welches die Inhaltstypen mit deren Feldern in semantische Klassen und Eigenschaften transformiert. Diese können um weitere benutzerdefinierte Ressourcen erweitert und unabhängig vom Ursprung untereinander verknüpft werden. Die entwickelte Website-Ontologie lässt sich zum einen innerhalb des CMS zur Annotierung der Inhaltstypen verwenden und zum anderen für eine externe Verarbeitung veröffentlichen. Im letzten Teil der Arbeit, werden die Implementierung in einem Beispiel getestet und Verbesserungsvorschläge für zukünftige Entwicklungen erarbeitet.

Abstract

Site ontologies are a special type of ontologies due to the close thematic relation to the website. In addition the website publishes the site ontology. The site ontology structures parts of the page contents as resources and deposits the knowledge with the aid of standardized languages of the Semantic Web. Thus humans and machines can handle it equally.

This thesis describes the process of creating a site ontology with a content management system (CMS) in a semi-automated way. Therefore, theoretical foundations are explained and the semantic CMS market is analyzed. Afterwards, the implemented module for *Drupal* CMS is presented, transforming the content types with their fields in semantic classes and properties. In addition, further custom resources can be arranged and link each other independently from the origin. On the one hand the site ontology is used for the annotation of the CMS content types and on the other hand the ontology is published for external processing. In the last part of the thesis, the implementation is tested on a sample page and suggestions for further improvements are made.

Kapitel 1

Einleitung

Das Semantic Web hat das Potential den Umgang mit Informationen nachhaltig zu verändern. Inhalte werden dazu nicht mehr wie bisher in Form von unstrukturierten Texten abgelegt, sondern einige Begriffe oder Abschnitte mit zusätzlichen Informationen angereichert. Damit können sowohl Menschen, als auch Maschinen die Bedeutung des Inhalts erfassen. Infolgedessen können Computer die Informationen besser aufbereiten und viel akkuratere Vorschläge unterbreiten als bisher.

Es bedarf ganz unterschiedlicher Technologien und Vorgehensweisen, damit die Vision des semantischen Netzes von Tim Berners-Lee Realität wird [4]. Eine davon ist die Entwicklung von Ontologien, die durch eine einheitliche Sprache für Mensch und Maschine zur Modellierung von Wissen das Rückgrat des Semantic Web darstellen.

Die Idee einer Website-Ontologie ist durch die Publikation von Corlosquet et al. entstanden [10], in der die Autoren eine Erweiterung für das Content Mangement System (CMS) *Drupal* vorstellen, die es dem Benutzer erlaubt die Inhalte semantisch zu annotieren und ein automatisiertes Vokabular aufzubauen. Darauf basierend soll das Vokabular aus Strukturinformationen des CMS auf eine höhere Stufe gestellt und zu einer Website-Ontologie weiterentwickelt werden.

Ein Ziel der Arbeit soll es sein, dem Benutzer in der gewohnten Arbeitsumgebung, also innerhalb des CMS, einen leichtgewichtigen Editor für die Erstellung einer Website-Ontologie zu bieten. Dieser soll den Benutzer durch eine teilautomatische Verarbeitung der Strukturinformationen unterstützen.

Durch die Nähe zum CMS ergibt sich zudem der Vorteil, dass die Ontologie thematisch mit der Website korreliert. Die erstellte Ontologie soll anschließend die Grundlage für die Strukturierung der eigenen Inhalte sein und kann darüber hinaus auch auf externen Seiten Verwendung finden.

Aufbau der Arbeit

Diese Arbeit gliedert sich in sieben Kapitel. Nach einer kurzen Einleitung werden im *Kapitel zwei* theoretische Grundlagen vorgestellt, die zu einem besseren Verständnis der verwendeten Techniken beitragen sollen. Im *Kapitel drei* werden die aktuellen Entwicklungen auf dem Gebiet der semantischen Content Management Systeme, sowie deren verwandte Vertreter, beleuchtet.

Das *Kapitel vier* erläutert kurz die Vorgehensweise bei der Erstellung und Integration von Ontologien und schlägt damit die Brücke zum Praxisteil dieser Arbeit. *Kapitel fünf* stellt ein selbst entwickeltes Modul zur Entwicklung von Website-Ontologien für das Content Management System Drupal vor und beschreibt den Verlauf von den Anforderungen über den Ansatz hin zur Implementierung eines Prototyps. Diese wird anschließend in einem Szenario vorgestellt.

Im *Kapitel sechs* wird die Implementierung in Form eines Vergleiches mit der bestehenden Erweiterung von Corlosquet et al. und einem Ontologie-Editor evaluiert sowie Verbesserungen und Vorschläge für eine weitere Entwicklung angebracht.

Abgeschlossen wird die Arbeit in *Kapitel sieben* mit einer Zusammenfassung und einem Ausblick mit Vorschlägen, an welchen Stellen die vorgestellte Implementierung noch verändert und weiterentwickelt werden kann.

Kapitel 2

Theoretische Grundlagen

In diesem Kapitel werden die grundlegenden Ideen des Semantic Webs skizziert und in weiterer Folge die zentralen Sprachen erläutert.

2.1 Das Semantic Web

Das *World Wide Web* (WWW) hat die Art wie Menschen miteinander kommunizieren grundlegend verändert. Die Geschwindigkeit, mit der Informationen ausgetauscht werden, befindet sich an der Grenze zur Echtzeit. Gleichzeitig wird die Menge der ausgetauschten Daten bis 2015 schätzungsweise 966 Exabytes pro Jahr betragen [9]. Daher ist es von entscheidender Bedeutung die relevanten Informationen herauszufiltern.

Bisher wird ein Großteil der publizierten Internetseiten durch Suchmaschinen besucht und indiziert, wodurch dem Benutzer eine Suche nach Schlagworten ermöglicht wird. In den meisten Fällen wird bei der Recherche eine große, für den Benutzer unrelevante Menge an Ergebnissen zurückgeliefert. Um nun die Präzision der Treffer zu erhöhen, bedarf es weiterer Mechanismen und Techniken.

Steve Bratt, seines Zeichens Chief Executive Officer des *World Wide Web Consortium*[1] (W3C), schlägt zur Lösung des beispielhaft angeführten Recherche-Problems zwei Wege vor [5].

Zum einen wird versucht den Computern mehr Intelligenz zu verleihen, so dass der Inhalt verstanden wird. Mit Natural Language Processing oder Bilderkennung existieren bereits Technologien in dieser Richtung. Auch Forschungen im Bereich der künstlichen Intelligenz (engl. Artificial Intelligence, AI) tragen einen Teil zum Verständnis bei. Jedoch lassen sich Anfragen an den Computer im Bereich der AI schwer formulieren, haben unter Umständen eine lange Laufzeit und geben schlussendlich keine oder unbefriedigende Ergebnisse zurück.

Eine weitere Möglichkeit wurde von Tim Berners-Lee, dem Erfinders des

[1] http://www.w3.org/

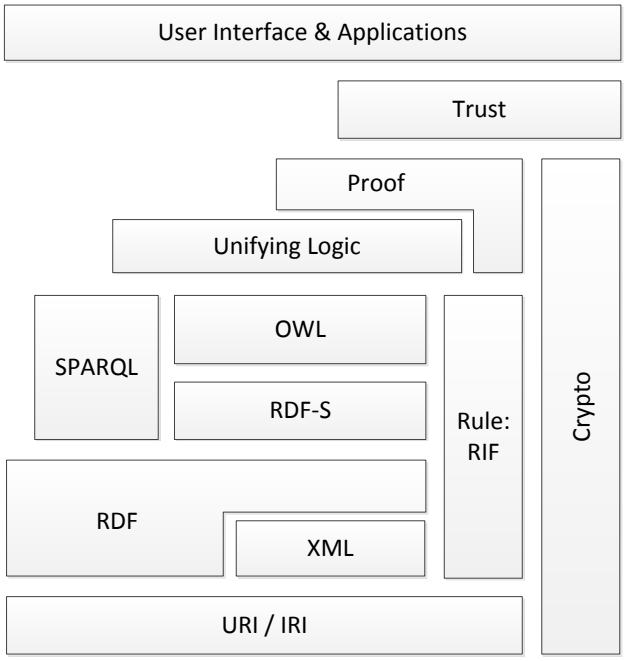

Abbildung 2.1: Semantic Web Stack des W3C [5].

WWW, mit der Idee des Semantic Webs vorgestellt. Berners-Lee schlägt eine Erweiterung des WWW vor, welche durch gezielte Anreicherung von Strukturinformationen eine maschinelle Verarbeitung zulässt [4]. Dazu soll die dezentrale Architektur des bisherigen WWW genutzt werden. Von der einfachen Anreicherung ausgehend können weitere logische Regeln definiert werden, wodurch Maschinen das automatische Schlussfolgern „erlernen“ und mit der Zeit auch einfache Fragen beantworten können. Als Gesamtbild entwirft Berners-Lee die Vision eines intelligenten Agenten, der den Benutzer bei der Bewältigung seines Alltags, durch Aufbereitung und Bereitstellung von relevanten Informationen, unterstützt [4].

2.2 Sprachen des Semantic Webs

Um die Vision eines maschinenlesbaren Webs zu verwirklichen, wurden verschiedene Sprachen entwickelt und standardisiert. Der *Semantic Web Stack* (auch Semantic Web Layer Cake) visualisiert die beteiligten Sprachen und Technologien in verschiedenen Schichten. Der Stack wurde ursprünglich von Berners-Lee entwickelt, wird jedoch mit fortschreitender Spezifikation der beteiligten Sprachen weiter überarbeitet. Daher existieren in der Literatur verschiedene Umsetzungen.

Die Abbildung 2.1 stellt die aktuelle Version des Stacks des W3C aus

dem Jahr 2007 dar [5]. Die Basis des Schichtenmodells bildet der internationale Zeichensatz Unicode zusammen mit *Uniform Resource Identifiern* (URI). Letztere bilden die Grundlage um Ressourcen im Internet zu identifizieren und kann im einfachsten Fall eine Internet- oder E-Mail-Adresse sein. Darüber hinaus existieren jedoch weitere URIs wie z. B. *Digital Object Identifier*[2] (DOI) für die Identifikation von digitalen Objekten oder geographischen Markierungen.

In der nächsthöheren Schicht definieren *Extensible Markup Language*[3] (XML) und *Resource Description Framework*[4] (RDF) die Syntax des Semantic Webs. Mit XML können strukturierte Dokumente von Mensch und Maschine gleichermaßen erstellt und gelesen werden. Daher hat sich XML in vielen Bereichen der Informatik als Austauschformat etabliert. Die Sprache *HyperText Markup Language*[5] (HTML), die zur Beschreibung von Internetseiten genutzt wird, basiert ebenso wie RDF auf XML. Aus diesem gemeinsamen Ursprung lässt sich ableiten, dass das Semantic Web, wie von Berners-Lee angedacht, wirklich das WWW erweitert und sich darin integriert. Für den weiteren Verlauf der Arbeit werden grundlegende Kenntnisse über XML vorausgesetzt und nur Besonderheiten in Bezug auf die Sprachen des Semantic Web aufgeführt.

Auf die Schichten RDF, RDFS und OWL soll an dieser Stelle nicht näher eingegangen werden, da sie ausführlicher in den nächsten Abschnitten erläutert werden.

Bei *SPARQL Protocol and RDF Query Language*[6] (kurz SPARQL, da rekursives Akronym) handelt es sich um eine Abfragesprache für RDF-Daten. Die benutzerdefinierten Abfragen können, ähnlich einer Datenbank-Abfrage, an spezielle Endpunkte mit SPARQL-Unterstützung gestellt werden. Als Ergebnis wird eine Menge an RDF-Daten zurückgegeben, die anschließend weiterverarbeitet werden kann [24].

Im Jahr 2010 ist das *Rule Interchange Format*[7] (RIF) zum Erstellen und Austausch von Regeln, ebenfalls vom W3C standardisiert worden. Mit Hilfe der aufgestellten Regeln lässt sich das Wissen kodieren und kann von Maschinen für automatische Schlussfolgerungen herangezogen werden.[8]

Die weiteren Schichten Unifiying Logic (einheitliche Logik), Proof (Nachweis), Trust (Vertrauen) und Crypto (Sicherheit) sind vom W3C bisher noch nicht spezifiziert. Sie werden jedoch benötigt, damit die Vision vom Semantic Web vollständig Realität wird.

---

[2] http://www.doi.org/

[3] http://www.w3.org/XML/

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

[5] http://www.w3.org/MarkUp/

[6] http://www.w3.org/TR/rdf-sparql-query/

[7] http://www.w3.org/2005/rules/wiki/RIF_Working_Group

[8] http://www.w3.org/2005/rules/wiki/RIF_FAQ, Kopie auf CD-ROM (Datei onlinequellen/2005_w3c_rif.pdf vom 13.09.2011).

Subjekt	Prädikat	Objekt
Ressource	Eigenschaft	Ressource
Ressource	Eigenschaft	Literal

Tabelle 2.1: Kombinationsmöglichkeiten eines RDF-Statements.

2.2.1 Resource Description Framework (RDF)

Das *Resource Description Framework*[9] (RDF) bildet die Grundlage für die aufbauenden Sprachen des Semantic Web Stack (s. Abb. 2.1). Die Konzeption erfolgte unabhängig von einer bestimmten Domäne, so dass es sich für vielfältige Einsatzgebiete eignet. Sollen eigene Terminologien entworfen werden, so können diese erst durch das RDF Schema (s. Kap. 2.2.2) definiert werden.

RDF beschreibt Informationen über Ressourcen, die über eine URI, in den häufigsten Fällen eine URL, identifiziert werden können. Dies ist unabhängig davon, ob die Ressource in digitaler Form vorliegt und die URI tatsächlich aufgerufen werden kann. Beispiele für Ressourcen sind Menschen, Organisationen, Bücher, Orte, Waren, usw.

Die Ressourcen wiederum werden durch Eigenschaften (engl. properties) näher beschrieben. Diese können z. B. Name, Alter, Farbe, Größe, usw. sein. Ebenso wie die Ressourcen werden auch die Eigenschaften über eine URI definiert.

Neben den Ressourcen und Eigenschaften existiert mit dem Literal noch ein weiterer Teil zur Beschreibung von Ressourcen. Ein Literal wird nicht durch eine URI identifiziert, sondern umfasst Datenwerte z. B. als Zahlen, Zeitangaben, Zeichenketten usw.

Zusammen gesetzt bilden die drei Teile, Subjekt, Prädikat, Objekt, ein RDF-Statement. Wobei ein Statement immer der Reihenfolge Subjekt, Prädikat und Objekt folgt. Wie in Tabelle 2.1 gezeigt, bestehen das Subjekt und Prädikat immer aus einer Ressource bzw. Eigenschaft und lediglich für das Objekt kann zwischen einer Ressource und einem Literal gewählt werden. Aus der definierten Dreiteilung leitet sich auch der Name „Tripel“ ab.

Die Abbildung 2.2 stellt die graphische Repräsentation mehrerer Statements für zwei Personen dar. Ein Statement ist in Form eines gerichteten Graphen vom Subjekt zum Objekt dargestellt. Die Ressourcen sind mit einer Ellipse, die Eigenschaften durch einen gerichteten Pfeil und ein Literal durch ein Rechteck symbolisiert. Eine detaillierte Erklärung des Aufbaus wird nachfolgend mit dem Programm 2.1 gegeben.

Die Aussagen über eine Ressource lassen sich in zwei verschiedenen Syntaxen ausdrücken. Zur Auswahl stehen das XML-Format oder die Notation

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

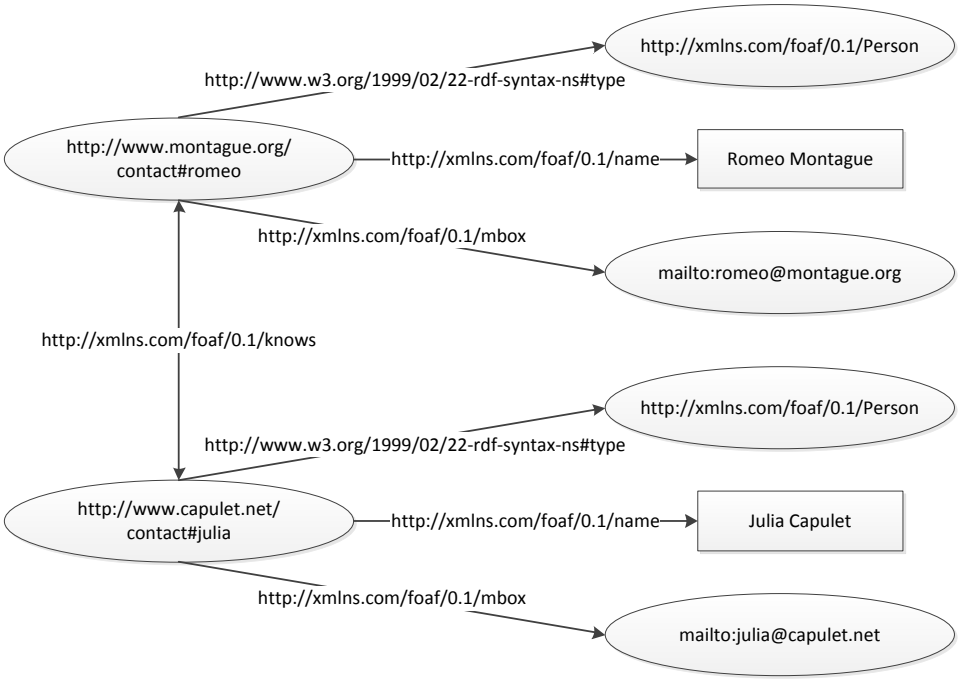

Abbildung 2.2: Graphische Repräsentation der RDF-Daten von zwei Personen.

```
1 <?xml version="1.0"?>
2 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3             xmlns:foaf="http://xmlns.com/foaf/0.1/">
4
5   <rdf:Description rdf:about="http://www.montague.org/contact#romeo">
6     <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person" />
7     <foaf:name>Romeo Montague</foaf:name>
8     <foaf:mbox rdf:resource="mailto:romeo@montague.org"/>
9     <foaf:knows rdf:resource="http://www.capulet.net/contact#julia"/>
10  </rdf:Description>
11
12  <foaf:Person rdf:about="http://www.capulet.net/contact#julia">
13    <foaf:name>Julia Capulet</foaf:name>
14    <foaf:mbox rdf:resource="mailto:julia@capulet.net"/>
15    <foaf:knows rdf:resource="http://www.montague.org/contact#romeo"/>
16  </foaf:Person>
17
18 </rdf:RDF>
```

Programm 2.1: RDF-Dokument mit Beschreibung von zwei Personen im XML-Format.

3 (N3) bzw. deren Untermenge Turtle (Terse RDF Triple Language) [3]. Die letzten beiden Repräsentationen zeichnen sich besonders durch eine kürzere Schreibweise aus, da die für XML notwendigen öffnenden und schließenden Tags entfallen. Im weiteren Verlauf der Arbeit wird jedoch die XML-Syntax verwendet.

Nachfolgend werden einige Besonderheiten von RDF anhand des Programms 2.1 vorgestellt. Das Programm zeigt die Implementierung der graphischen Repräsentation aus Abbildung 2.2 im XML-Format.

Jedes RDF-Dokument beginnt mit der standardmäßigen XML-Deklaration (Zeile 1) und dem umschließenden Wurzelknoten in Zeile 2. Der Wurzelknoten enthält die XML-Namensräume für RDF und *Friend-Of-A-Friend* (FOAF). FOAF ist ein Vokabular zur Beschreibung von Personen, deren Eigenschaften, sowie Beziehungen zwischen Personen [7]. In dem gewählten Beispiel sind die beiden Personen Romeo und Julia in RDF beschrieben. Mittels FOAF wurde beiden Personen ein Name und eine E-Mail-Adresse zugeordnet. Zudem wurde die Aussage getroffen, dass beide einander kennen.

Mit `rdf:Description` und `rdf:about` in Zeile 5 wird die Beschreibung der Ressource eingeleitet. Dabei wird `rdf:about` eine URI zugewiesen, wodurch die Ressource eindeutig identifiziert wird. Die nachfolgende Zeile 6 legt fest, dass es sich bei der beschreibenden Ressource um eine Person aus dem FOAF-Vokabular handelt. Die Verknüpfung wird mittels `rdf:resource` durch eine URI gesetzt. Die Zeile 12 stellt eine verkürzte Schreibweise für `rdf:Description` und `rdf:type` dar, bei der die verknüpfte Typ-Ressource direkt als XML-Tag geschrieben wird. Beide Varianten können äquivalent verwendet werden.

Die Spezifikation sieht über die vorgestellten RDF-Tags hinaus weitere Tags, z. B. für sortierte und unsortierte Listen, vor, die an dieser Stelle nicht besprochen werden.

2.2.2 Resource Description Framework Schema (RDFS)

Mit RDF wurden bisher Ressourcen beschrieben, die als Individuen verstanden werden können. Mit dem *Resource Description Framework Schema*[10] (RDFS) lassen sich abstrakte oder generische Konstrukte beschreiben [6].

RDFS ist zugleich eine Erweiterung und Grundlage der Beschreibungssprache RDF und definiert ein standardisiertes Vokabular, das auch zur Spezifizierung von RDF verwendet wird. Darauf basierend lassen sich benutzerdefinierte Vokabulare implementieren [1]. Grundlegende Merkmale von RDF, wie die Konstruktion von Aussagen in Form von Tripeln, finden in RDFS ebenfalls Verwendung.

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

Element	*Beschreibung*	Super-Klasse	Typisierung
rdfs:Resource	Basis für Ressourcen	rdfs:Resource	rdfs:Class
rdfs:Class	Basis für Klassen	rdfs:Resource	rdfs:Class
rdf:Property	Basis für Eigenschaften	rdfs:Resource	rdfs:Class
rdfs:Literal	Literale	rdfs:Resource	rdfs:Class
rdfs:Datatype	Datentypen	rdfs:Class	rdfs:Class
rdfs:Container	Container	rdfs:Resource	rdfs:Class

Tabelle 2.2: Ausgewählte Klassen des RDF- und RDFS-Vokabulars nach den Spezifikationen des W3C [6].

RDFS-Klassen

Die Tabelle 2.2 wurde auf Basis der RDF-Spezifikation erstellt [6] und listet ausgewählte Elemente des RDF- und RDFS-Vokabulars mit deren Super-Klassen und definierter Typisierung auf.

Die Basis-Klasse `rdfs:Resource` ist vergleichbar mit der Java-Klasse `Object`, die als Wurzel für alle abgeleiteten Klassen der Vererbungshierarchie steht und von sich selbst erbt. Die Typisierung der Ressourcen nach Klassen und Eigenschaften erfolgt mit `rdfs:Class` und `rdf:Property`, wobei zu beachten ist, dass sich `rdfs:Class` selbst typisiert und `rdf:Property` ein Bestandteil von RDF und nicht RDFS ist [6].

Mit `rdfs:Datatype` können benutzerdefinierte Datentypen spezifiziert werden. Eine Besonderheit ist, dass die neuen Typen mit `rdfs:Datatype` eingeteilt, jedoch von der Klasse `rdf:Literal` abgeleitet sind [13]. Die Klasse `rdf:Literal` definiert die Verwendung von Literalen bzw. Zeichenketten.

RDFS-Eigenschaften

Die Tabelle 2.3 ist auf Basis der RDF-Spezifikation [6] angefertigt und stellt eine Auswahl an Prädikaten des RDF- und RDFS-Vokabulars mit deren Einschränkung für Subjekte und Objekte von RDF Tripel dar.

Das Prädikat `rdf:type` ist Bestandteil des RDF-Vokabulars und verknüpft Ressourcen mit einer Klasse. Durch die Zuweisung wird die Ressource automatisch zu einer Instanz der Klasse.

Durch die Eigenschaften `rdfs:subClassOf` und `rdfs:subPropertyOf` lassen sich Hierarchien von Klassen bzw. Eigenschaften modellieren. Jedes Subjekt kann durch multiple Verwendung des Prädikates von mehreren Super-Klassen bzw. Super-Eigenschaften abgeleitet sein [6].

Ein Prädikat kann standardmäßig auf jedes Subjekt und mit jedem Objekt verwendet werden. In vielen Fällen ist es sinnvoll, das Subjekt und das Objekt eines Tripels auf eine bestimmte Klasse einzuschränken und so einen

Element	*Beschreibung*	*rdfs:domain*	*rdfs:range*
rdf:type	Typisierung der Ressource	rdfs:Resource	rdfs:Class
rdfs:subClassOf	Angabe der Super-Klasse	rdfs:Class	rdfs:Class
rdfs:subPropertyOf	Angabe der Super-Eigenschaft	rdf:Property	rdf:Property
rdfs:range	Einschränkung für Subjekte	rdf:Property	rdfs:Class
rdfs:domain	Einschränkung für Objekte	rdf:Property	rdfs:Class
rdfs:label	Menschenlesbares Label	rdfs:Resource	rdfs:Literal
rdfs:comment	Menschenlesbarer Kommentar	rdfs:Resource	rdfs:Literal
rdfs:seeAlso	Weitere Informationen	rdfs:Resource	rdfs:Resource
rdfs:isDefinedBy	Definition	rdfs:Resource	rdfs:Resource

Tabelle 2.3: Ausgewählte Prädikate des RDF- und RDFS-Vokabulars nach den Spezifikationen des W3C [6].

```
1 <rdf:Property rdf:ID="isWritten">
2   <rdfs:domain rdf:resource="#Book"/>
3   <rdfs:range rdf:resource="#Author"/>
4 </rdf:Property>
```

Programm 2.2: Beschränkung der Eigenschaft „isWritten“ anhand von `rdfs:domain` und `rdfs:range`.

Bereich vorzugeben, in dem das Prädikat genutzt werden darf. RDFS stellt für diesen Fall die Elemente `rdfs:domain` und `rdfs:range` bereit, für die eine Klasse angegeben wird [6]. Die Einschränkung der angegebenen Klassen schließt eine Verwendung von abgeleiteten Subklassen ein. Das Programm 2.2 veranschaulicht das Prinzip anhand der Eigenschaft „isWritten“, welche definiert, dass nur Bücher von Autoren geschrieben werden dürfen. Klassen eines anderen Typs sind damit für das Subjekt bzw. Objekt unzulässig.

Die beiden Eigenschaften `rdfs:label` und `rdfs:comment` erklären eine Ressource durch das Anbringen einer Zeichenkette. Die Beschreibung ist vorzugsweise für Menschen gedacht, wobei das Label auch von einigen Programmen für die graphische Repräsentation des Tripels genutzt wird. Der Kommentar enthält gewöhnlicherweise eine detaillierte Erklärung zur Res-

source.

Die Eigenschaft `rdfs:seeAlso` verweist vom Subjekt auf eine zusätzliche Beschreibung in Form einer weiteren Ressource (Objekt). Die davon abgeleitete Eigenschaft `rdfs:isDefinedBy` beschreibt spezieller, wo eine (erweiterte) Definition des Subjekts abgelegt ist. Bei der verknüpften Ressource (Objekt) handelt es sich überwiegend um ein weiteres RDFS-Dokument [6].

Verwendet man die Menge der erläuterten Eigenschaften zur Erstellung eines Vokabulars, können aus dem Datenbestand bereits einfache Inferenzalgorithmen angewendet werden [13]. Dadurch kann z. B. geschlussfolgert werden, so es definiert ist, dass es sich bei der Person Romeo um einen Menschen handelt.

Grenzen von RDFS

Die Einfachheit des Vokabulars setzt auch Grenzen. So ist es mit der Semantik von RDFS nicht möglich eine Ressource auszuschließen oder eine Aussage zu negieren [13]. Möchte man beispielsweise festlegen, dass Romeo Julia *nicht* liebt, dann wäre das mit RDFS nur über den Umweg einer Eigenschaft `ex:nichtVerliebtIn` möglich.

2.2.3 Web Ontology Language (OWL)

Die *Web Ontology Language*[11] (OWL) erweitert die einfache Semantik von RDFS um eine standardisierte Beschreibungssprache für Ontologien. Das Wort Ontologie entstammt aus dem Griechischen und ist eine Disziplin der Philosophie, die sich mit der Beschreibung von Entitäten und deren Beziehungen untereinander beschäftigt. In der Informatik wird der Begriff Ontologie übernommen und dahingehend erweitert, dass ein Modell der Wirklichkeit formuliert wird. Die erstellten Entitäten werden über die Beziehungen hinaus mit Inferenz- und Integritätsregeln angereichert, die dem logischen Schlussfolgern und der Gewährleistung der Gültigkeit dienen [1]. Durch logisches Schlussfolgern kann auf implizites Wissen der Ontologie zugegriffen werden [13].

Ausprägungen von OWL

OWL ist kompatibel zu RDF und RDFS und verwendet Teile der Vokabulare wieder. Der Standardisierung von OWL ist eine Initiative vorausgegangen, die aus den beiden Vorgängern DAML-ONT[12] und OIL[13] eine gemeinsame Sprache DAML+OIL[14] entworfen hat [13]. Nachdem das W3C die Leitung zur Definition der Ontologiesprache übernommen hat, wurde die Sprache

[11] http://www.w3.org/2004/OWL/
[12] http://www.daml.org/2000/10/daml-ont.html
[13] http://www.ontoknowledge.org/oil/
[14] http://www.daml.org/2001/03/daml+oil-index.hmtl

OWL im Jahr 2004 als offizieller Standard veröffentlicht. Seit 2009 empfiehlt das W3C die überarbeitete und erweiterte zweite Version *OWL 2*[15] zu verwenden, die mit der ersten Version kompatibel ist und um einige Funktionen erweitert wurde [21].

Die Spezifikation sieht drei Ausprägungen von OWL mit differenzierten Einschränkungen und Unterstützungen vor, die nach Hitzler et al. nachfolgende Merkmale aufweisen [13].

OWL Full Die umfassendste und mächtigste Variante ist OWL Full, welche die Teilsprachen OWL DL und OWL Lite enthält. Darüber hinaus ist jedes RDF- bzw. RDFS-Dokument ein gültiges OWL Full, da sich die komplette Syntax abbilden lässt. OWL Full hat jedoch den Nachteil, dass es bei Schlussfolgerungen und Anfragen an die Wissensbasis keine Entscheidung bzw. kein Ergebnis liefert und damit unentscheidbar ist.

OWL DL In der zweiten Ausprägung OWL DL (engl. Description Logic) können Schlussfolgerungen durch einige Einschränkungen des Vokabulars gegenüber OWL Full effizient gelöst werden. Daher gilt OWL DL als entscheidbar. Die Ausprägung ist die am weitesten verbreitetste und wird auch von gängigen Programmen wie Ontologie-Editoren unterstützt.

OWL Lite Die dritte Ausprägung OWL Lite ist wiederum nur eine Teilmenge von OWL DL und beschneidet die Sprache um weitere Elemente. Sie ist der ausdruckschwächste OWL-Vertreter, jedoch wie OWL DL entscheidbar.

Über die drei vorangestellten Ausprägungen hinaus, existieren in der Spezifikation für OWL 2, drei weitere Teilsprachen: OWL 2 EL (polynomialzeit Algorithmen für Standard-Schlussfolgerungen), OWL 2 QL (für SQL-ähnliche Anfragen) und OWL 2 RL (polynomialzeit Algorithmen mit regelbasierten Datenbanken, die direkt mit den Tripeln arbeiten) [21]. Diese spielen für die folgenden Grundlagen lediglich eine untergeordnete Rolle.

Offene-Welt-Annahme

Wie bereits im Kapitel 2.1 beschrieben, wurde das Semantic Web von Tim Berners-Lee im Sinne des WWW dezentral geplant. Dadurch ergeben sich für das logische Schlussfolgern in Ontologien eine Reihe von Problemen. So kann beispielsweise nicht davon ausgegangen werden, dass (a) eine verweisende Ressource einer Ontologie zum Zeitpunkt der Schlussfolgerung verfügbar ist oder (b) an anderen Orten im Semantic Web weitere Informationen zur Ressource existieren. Daher wurde die *Offene-Welt-Annahme* (OWA, engl. Open World Assumption) aus der Wissensrepräsentation übernommen, welche davon ausgeht, dass eine Wissensbasis permanent potentiell unvollständig ist [13]. Nimmt man z. B. das Statement „Romeo Montague lebt in der

[15] http://www.w3.org/TR/owl2-overview/

Stadt Verona“ an, so kann die Frage, ob Julia Capulet ebenfalls in der Stadt Verona lebt, nach der OWA nicht entschieden werden, da noch weitere Fakten existieren können. Erst wenn ein Statement (gleichgültig ob positiv oder negativ) zu Julias Wohnort bekannt ist, kann eine Aussage getroffen werden. Der OWA steht die Geschlossene-Welt-Annahme (engl. Closed World Assumption) gegenüber, die z. B. aus Datenbanken bekannt ist und annimmt, dass alle relevanten Fakten in der Wissensbasis vorhanden sind [13]. Stellt man die gleiche Frage an ein geschlossenes System, so erhält man die Antwort, dass Julia nicht in Verona lebt.

OWL-Klassen und -Eigenschaften

OWL bringt, wie RDF bzw. RDFS, ebenfalls ein definiertes Vokabular an Klassen und Eigenschaften mit. Die Tabelle 2.4 ist auf Basis der OWL-Spezifikation [26] angefertigt und stellt einige Elemente vor, die im Folgenden näher beschrieben werden.

Mit dem Element `owl:Ontology` wird der Ontologie-Kopf eingeleitet, der einige Metadaten des OWL-Dokumentes beschreibt. Metadaten können der Titel (`rdfs:label`), eine Beschreibung (`rdf:comment`), sowie der Import von weiteren verwendeten Ontologien (`owl:import`) oder ein Verweis auf vorherige Versionen (`owl:priorVersion`) der Ontologie sein [26].

Im Regelfall ist eine Ontologie nicht komplett und wird fortlaufend verfeinert. Für den produktiven Einsatz von Ontologien ist es daher von Bedeutung die Änderungen transparent für Mensch und Maschine nachzuvollziehen, um so Inkonsistenzen vorzubeugen. Folglich wurden für die Versionierung von Ontologien in OWL eine ganze Reihe an Elementen definiert, die den Umgang erleichtern. Mit den Elementen `owl:DeprecatedClass` und `owl:DeprecatedProperty` können Klassen und Eigenschaften als veraltet markiert werden und auf eine Änderung in der nächsten Version hinweisen. Mit `owl:incompatibleWith` und `owl:backwardCompatibleWith` kann gekennzeichnet werden, dass ein Element zu einem anderen Element inkompatibel oder im Gegenteil abwärtskompatibel ist [13].

Wie bereits ausgeführt, kann mit RDFS nicht ausgedrückt werden, dass eine Ressource nicht existiert. In OWL wiederum wird das Problem über die beiden vordefinierten Klassen `owl:Thing` und `owl:Nothing` gelöst. Die beiden Klassen sind Superklassen für jede abgeleitete OWL-Klasse, wobei `owl:Thing` alle Individuen enthält und `owl:Nothing` als leer definiert ist [2].

OWL sieht ebenfalls eine Trennung zwischen Klassen und Eigenschaften vor. Klassen werden mit `owl:Class` ausgezeichnet und dienen als Grundlage zur Erstellung von Instanzen. Die Instanzierung funktioniert identisch zur Vorgehensweise in RDFS mit `rdf:Description` und `rdf:type`. Mit der Eigenschaft `owl:disjointWith` können Klassen von einander unabhängig modelliert und verwendet werden. Die Disjunktion wird automatisch auf abgeleitete Klassen vererbt, so dass (je nach Anatomie der Ontologie) lediglich

Element	*Beschreibung*
owl:backwardCompatibleWith	Angabe zur Abwärtskompatibilität
owl:cardinality	Festlegen der Kardinalität
owl:Class	Basis für Klassen
owl:DatatypeProperty	Eigenschaft mit Datentyp als Wertebereich
owl:DeprecatedClass	Veraltete Klasse
owl:DeprecatedProperty	Veraltete Eigenschaft
owl:differentFrom	Differenzierung zweier Elemente
owl:disjointWith	Disjunktion zweier Klassen
owl:equivalentClass	Angabe einer äquivalenten Klasse
owl:equivalentProperty	Angabe einer äquivalenten Eigenschaft
owl:FunctionalProperty	Funktionale Eigenschaft
owl:imports	Import einer fremden Ontologie
owl:incompatibleWith	Angabe zur Inkompatibilität
owl:InverseFunctionalProperty	Invers funktionale Eigenschaft
owl:inverseOf	Angabe der inversen Eigenschaft
owl:maxCardinality	Obergrenze der Kardinalität
owl:minCardinality	Untergrenze der Kardinalität
owl:Nothing	Leere Klasse
owl:ObjectProperty	Eigenschaft mit Objektverweis als Wertebereich
owl:onProperty	Einschränkungen einer Rolle in Verbindung mit `owl:Restriction`
owl:Ontology	Container für Ontologie-Metadaten
owl:priorVersion	Vorherige Ontologie-Version
owl:Restriction	Einschränkungen einer Rolle in Verbindung mit `owl:onProperty`
owl:sameAs	Äquivalenz zweier Instanzen
owl:SymmetricProperty	Symmetrische Eigenschaft
owl:Thing	Klasse von Allem
owl:TransitiveProperty	Transitive Eigenschaft
owl:versionInfo	Angabe zur Ontologie-Version

Tabelle 2.4: Ausgewählte Elemente des OWL-Vokabulars nach den Spezifikationen des W3C [26].

Super-Klassen disjunktiv verknüpft werden müssen [2].

Die Eigenschaften, in OWL auch Rollen genannt, werden in OWL differenziert behandelt und mit den beiden Elementen `owl:ObjectProperty` oder `owl:DatatypeProperty` beschrieben. Das Programm 2.3 veranschaulicht den Unterschied. Die Objekt-Eigenschaft `owl:ObjectProperty` ist ver-

```
1 <owl:ObjectProperty rdf:ID="isWritten">
2   <rdfs:domain rdf:resource="#Book"/>
3   <rdfs:range rdf:resource="#Author"/>
4 </owl:ObjectProperty>
5
6 <owl:DatatypeProperty rdf:ID="title">
7   <rdfs:domain rdf:resource="#Book"/>
8   <rdfs:range rdf:resource="&xsd;string"/>
9 </owl:DatatypeProperty>
```

Programm 2.3: Objekt- und Datentyp-Eigenschaften in OWL, mit deren Eingrenzungen durch `rdfs:domain` und `rdfs:range`.

gleichbar mit `rdf:Property` und schränkt den Objekt-Bereich von „isWritten“ mit `rdfs:range` auf einen Autor ein. Für die Datentyp-Eigenschaft `owl:DatatypeProperty` wird im Objekt-Bereich des Buchtitels ein String als XML-Datentyp erwartet.

Das Programm 2.4 stellt eine Reihe von OWL-Eigenschaften dar. Mit diesen ist es möglich die Beziehungen differenzierter zu beschreiben, als es mit der RDFS-Eigenschaft `rdfs:subPropertyOf` möglich ist. Die Eigenschaft `owl:inverseOf` in Zeile 1 definiert, dass zu einer Rolle eine entgegengesetzte Rolle existiert [1]. Ab Zeile 5 wird eine funktionale Eigenschaft (`owl:FunctionalProperty`) „wife“ angelegt, die genau eine eindeutig zu identifizierende Ehefrau zu einem Ehemann zu lässt. Mehrere Frauen wären in diesem Beispiel unzulässig. Das Beispiel einer invers funktionalen Eigenschaft (`owl:InverseFunctionalProperty`) ab Zeile 11 beschreibt, dass für einen Menschen mit einer biologischen Mutter genau eine eindeutig zu identifizierende Frau existieren muss [2]. Damit ist die Funktionsweise ähnlich den Key-Notationen bei der Datenbankentwicklung. Die transitive Eigenschaft (`owl:TransitiveProperty`) ab Zeile 16 beschreibt, dass ein Teil aus weiteren Teilen bestehen kann. Wenn folglich die Karosserie Bestandteil eines Autos ist und die Autotür wiederum ein Teil der Karosserie ist, so kann aufgrund der Transitivität geschlussfolgert werden, dass auch die Autotür ein Bestandteil des Autos ist. Die letzte Eigenschaft ist `owl:SymmetricProperty` die ab Zeile 21 deklariert ist. Diese definiert, dass die Eigenschaften in beide Richtungen existieren. Im Beispiel gesprochen, dass die Liebe zwischen zwei Menschen im Idealfall von beiden Seiten erwidert wird und nicht einseitig ist.

Neben der Modellierung von Beziehungen zwischen Eigenschaften, lassen sich Rollen mit `owl:Restriction` unter Angabe der betreffenden Eigenschaft mit `owl:onProperty` einschränken. Zur Auswahl stehen die folgenden Restriktionen, die nach Smith et al. beschrieben wurden [26]:

`owl:allValuesFrom` Alle Werte müssen vom Typ der angegebenen Klasse

```
<owl:ObjectProperty rdf:ID="hasChild">
  <owl:inverseOf rdf:resource="#hasParent"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="wife">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
  <rdfs:domain rdf:resource="#Man" />
  <rdfs:range rdf:resource="#Woman" />
</owl:ObjectProperty>

<owl:InverseFunctionalProperty rdf:ID="biologicalMotherOf">
  <rdfs:domain rdf:resource="#Woman"/>
  <rdfs:range rdf:resource="#Human"/>
</owl:InverseFunctionalProperty>

<owl:TransitiveProperty rdf:ID="isPartOf">
  <rdfs:domain rdf:resource="#Part"/>
  <rdfs:range rdf:resource="#Part"/>
</owl:TransitiveProperty>

<owl:SymmetricProperty rdf:ID="loves">
  <rdfs:domain rdf:resource="#Human"/>
  <rdfs:range rdf:resource="#Human"/>
</owl:SymmetricProperty>
```

Programm 2.4: OWL-Eigenschaften zum Spezifizieren von Beziehungen.

sein.

`owl:someValuesFrom` Mindestens ein Wert muss vom Typ der angegebenen Klasse sein.

`owl:hasValue` Der Wert muss genau vom Typ der angegebenen Klasse sein.

`owl:cardinality` Angabe eines positiven Integers, um die genaue Anzahl der Werte einzuschränken.

`owl:minCardinality` Angabe eines positiven Integers, um die Untergrenze für die Anzahl der Werte festzulegen.

`owl:maxCardinality` Angabe eines positiven Integers, um die Obergrenze für die Anzahl der Werte festzulegen.

Die definierten Aussagen über eine Ressource können sich über das gesamte Semantic Web verteilen. Damit ein Netz an Informationen zu einer Ressource gebildet wird, stellt OWL einige Elemente bereit, um zwei Ressourcen zu verknüpfen. In der Literatur wird dieser Vorgang auch Ontologie-Mapping genannt, welches über die einzelnen Ressourcen hinaus, das Verknüpfen einer oder mehrerer Ontologien miteinander beschreibt. In OWL können die Instanzen zweier Klassen mit `owl:equivalentClass` als äquivalent definiert werden. Man beachte, dass in OWL DL die Klassen eine Samm-

lung von Instanzen kennzeichnet und nicht die Instanzen selber [26]. Das Element `owl:equivalentProperty` bildet den gleichen Mechanismus für Eigenschaften ab. Neben den Klassen können auch Instanzen mit `owl:sameAs` als äquivalent markiert werden. Um eine Unterscheidung kenntlich zu machen, wird `owl:differentFrom` verwendet. Es ist zu beachten, dass in OWL Full Klassen, Rollen und Instanzen gleich behandelt werden und damit das Element `owl:sameAs` sowohl für Instanzen als auch für Klassen verwendet werden kann. Die Aufhebung der Trennung ist ein Grund, warum Aussagen in OWL Full für Inferenzmaschinen nicht entscheidbar sind [13].

Die vorgestellten Elemente stellen lediglich einen Ausschnitt des OWL-Vokabulars dar. Über die angesprochenen Charakteristiken der Elemente hinaus existieren eine ganze Reihe an Sonderregelungen, die zum Teil durch die verschiedenen Ausprägungen (OWL Lite, OWL DL, usw.) entstehen. An die Erstellung von Ontologien grenzt u. a. das umfassende Feld des automatisierten Schlussfolgerns. Eine detaillierte Betrachtung würde den Rahmen der Arbeit übersteigen und ist somit nicht Gegenstand dieser Arbeit.

Kapitel 3

State of the Art

Nachdem die theoretischen Grundbegriffe im vorangegangen Kapitel ausgeführt wurden, werden nun aktuelle Entwicklungen im Bereich der semantischen Verwaltungen dargelegt.

Als erstes soll das Spezialgebiet der digitalen Bibliotheken betrachtet und die Definition von Content Mangement Systemen erörtert werden. Daran anschließend wird die semantische Umsetzung im Digital Asset Management und den bekanntesten Content Management Systemen ausgeführt.

3.1 Semantische digitale Bibliotheken

Die digitale Bibliothek (engl. Digital Library, DL), teilweise auch virtuelle Bibliothek genannt, wird in verschiedenen Kontexten genutzt und jeweils anders definiert.

Die *Association of Research Libraries* hat 1995 eine Definition für DL ausgearbeitet.[1] Demnach ist die DL lediglich ein Teil eines ganzen Verbundes, durch die mehrere DLs miteinander verknüpft werden. Die Benutzer können durch den Verbund mit einer einheitlichen Schnittstelle auf den verschiedenen DLs zugreifen. Darüber hinaus enthalten DLs teilweise oder ganz Daten, die lediglich in digitaler Form verfügbar sind und beispielsweise nicht in gedruckter Form existieren.

Candela et al. erweitern die vorangehende Definition mit ihrem Referenzmodell für digitale Bibliotheken [8]. Sie identifizieren und beschreiben die folgenden drei Schichten einer DL:

- Digital Library (DL) ist eine Organisation oder ein Dienst, welcher digitale Inhalte langfristig aufbewahrt und Benutzern zugänglich macht.
- Digital Library System (DLS) ist ein Verbundsystem aus DL. Benutzer verwenden das DLS um auf DL zu zugreifen.

[1] http://www.arl.org/resources/pubs/mmproceedings/126mmappen2.shtml, Kopie auf CD-ROM (Datei onlinequellen/1995_arl_definition-of-a-digital-library.pdf vom 13.09.2011).

- Digital Library Management System (DLMS) ermöglicht die Administration und Erweiterung von DLS.

Eine Bibliothek lebt von der Suche. Sie ermöglicht es erst, die enthaltenen Bücher, Medien oder allgemeinen Informationen für den Benutzer auffind- und damit konsumierbar zu machen. Folglich ist es von enormer Relevanz die Suche nach Informationen präziser und effizienter zu gestalten. Um dieses Ziel zu erreichen, werden die Ressourcen in einer semantischen DL, zusätzlich zu den herkömmlichen Metadaten, mit weiteren semantischen Informationen versehen und abgelegt. Die Suche ist dann beispielsweise in der Lage Ressourcen zu finden, auch wenn das Wort oder die exakte Phrase nicht bekannt ist.

Zur Organisation von Ressourcen in einer Bibliothek wird bisher auf Standards, wie *MARC21* in verschiedenen Ausprägungen, *Dublin Core* und *BibTeX* zurückgegriffen. Alle besitzen die Gemeinsamkeit, dass die Daten in einer strukturierten, maschinenlesbaren Form abgelegt werden und damit den Austausch von Daten fördern.

Die *MarcOnt Initiative*[2] hat es sich zur Aufgabe gemacht, eine einheitliche semantische Ontologie auf Basis der drei genannten Standards zu entwerfen [27]. Sie fördert damit die Interoperabilität zwischen verschiedenen DL und bereitet den Weg für ein Verbundsystem wie es Candela et al. identifiziert haben. Der Einsatz der Ontologie wird in Wechselwirkung mit der digitalen Bibliothek *JeromeDL*[3] verbessert und weiterentwickelt.

JeromeDL ist ein Vertreter der semantischen digitalen Bibliothek und wird hauptsächlich vom *Digital Enterprise Research Institute* (DERI) entwickelt. Neben MarcOnt erweitert auch die eigene JeromeDL Ontologie die verwalteten Ressourcen um semantische Informationen. Des Weiteren implementiert JeromeDL die FOAFRealm[4] Code-Bibliothek, mit der Zugriffe für verteilte Benutzerprofile auf Basis des Friend-Of-A-Friend (FOAF) Vokabulars festgelegt werden [17].

3.2 Definition von Content Management Systemen

Ein Content Management System (CM-System, CMS) bezeichnet ein Programm, das zum Verwalten von Inhalten verwendet wird. Das Gebiet der CMS gliedert sich in Web Content Management Systemen (WCM-Systeme, WCMS) für die Verwendung im Internet und Enterprise Content Management Systemen (ECM-Systeme, ECMS) für den Einsatz in Unternehmen [12]. Nach Angaben der Association for Information and Image Management (AIIM), dem Dachverband für ECM-Systeme, schließt das ECMS den

[2] http://www.marcont.org/
[3] http://www.jeromedl.org/
[4] http://www.foafrealm.org/

Bereich WCMS ebenfalls als Sub-Komponente ein.[5]

Im Rahmen der Arbeit soll nicht weiter auf ECMS eingegangen und stattdessen der Bereich WCMS näher betrachtet werden. Im weiteren Verlauf der Arbeit verwendet der Autor daher den Begriff CMS synonym zum Begriff WCMS.

Die Kernfunktionalität eines CMS ist es, den Inhalt von Layout und Strukturinformationen zu trennen und damit erst die Verwaltung und Wiederverwendung von Inhalten in anderen Kontexten zu ermöglichen [18]. Inhalte können je nach Umfang und Art des Systems aus Texten, Bildern, Audio- oder Video-Dateien bestehen und untereinander kombiniert werden. Das Layout wiederum definiert, wie der Inhalt präsentiert wird. Die Struktur setzt sich aus Klassifikation und Metainformationen über den Inhalt zusammen und ist für den Benutzer nicht oder nur teilweise einsehbar. Die Informationen dienen dem CMS vorrangig zur Verwaltung der Inhalte.

Nach Kampffmeyer [14] bietet ein CMS klassischerweise die folgenden Funktionalitäten:

- Erstellung und Bearbeitung von Inhalten
- Verwaltung und Organisation von Inhalten
- Präsentation und Distribution von Inhalten
- (Rechte-)Kontrolle und Versionierung von Inhalten
- Personalisierung und Individualisierung von Inhalten

Die Anzahl der Kategorien macht deutlich, dass es einen großen Spielraum hinsichtlich der Funktionalitäten bei den CM-Systemen gibt. So existiert ein breit gefächertes Spektrum, das von einfachen Lösungen zum Pflegen der privaten Homepage bis zu komplexen Systemen, die auf den Einsatz in größeren Unternehmen ausgelegt sind, reicht.[6]

3.3 Semantisches Digital Asset Management

Eine Unterkategorie des CMS stellt das Digital Asset Management (DAM) dar. Das System ist auf das Verwalten von Assets spezialisiert. Bei Assets kann es sich sowohl um einfache Textdokumente als auch um Bilder, Musik oder Videos handeln. Als Kernfunktionalitäten können die einfache Eingliederung in den Arbeitsablauf, das automatische Konvertieren und Bearbeiten der Assets, sowie ein feingranulares Rechtemanagement identifiziert werden.

Mit *Fedora Commons*[7] und *DSpace*[8] existieren zwei generische Open-Source Lösungen, die je nach Konfiguration auch an der Grenze zum Do-

[5] http://www.aiim.org/What-is-ECM-Enterprise-Content-Management, Kopie auf CD-ROM (Datei onlinequellen/2011_aiim_what-is-ecm.pdf vom 13.09.2011).

[6] Liste von bekannten Open-Source und proprietären CMS: http://en.wikipedia.org/wiki/List_of_content_management_systems

[7] http://www.fedora-commons.org/

[8] http://www.dspace.org/

kumenten Management bzw. zur digitalen Bibliothek einzuordnen sind. Für beide Systeme können je nach Verwendungszweck verschiedene Objekttypen definiert werden. Des Weiteren zeichnet beide Systeme eine ausgereifte Versionierung und eine hohe Langzeitverfügbarkeit der Ressourcen aus.

Kinner stellt in seiner Arbeit eine Lösung vor, wie DSpace mit Hilfe von Dublin Core, Harmony ABC und DSpace History um ein RDF Schema erweitert und so die Versionierung der DSpace Objekte interoperabel gestaltet werden können [16].

Obwohl, im Vergleich zur digitalen Bibliothek, mit der Dublin Core Ontologie bereits ähnliche Möglichkeiten zur semantischen Anreicherung für generische Metadaten existieren, ist über das angeführte Beispiel hinaus eine semantische Unterstützung im DAM Bereich kaum bis gar nicht verbreitet.

3.4 Semantische Content Management Systeme

Um einen Überblick über semantische Content Management Systeme zu geben, sollen die bekanntesten WCMS anhand ihrer Verbreitung ausgewählt werden.

Shreves analysiert in einem jährlichen Report die Marktverbreitung von Open-Source CMS. Für 2010 diagnostiziert er durch eine Umfrage unter 4000 Teilnehmern, dass *Joomla!*[9], *WordPress*[10] und *Drupal*[11] die höchste Anzahl an Installationen vorweisen können [25].

Demgegenüber analysiert W3Techs fortlaufend die Marktanteile der populärsten CMS anhand von technischen Daten der ersten Million Seiten im Alexa Ranking. Die Untersuchung bewertet WordPress mit über 50% Marktanteil, gefolgt von Joomla! mit 10% und Drupal mit knapp 6% [30]. Damit unterscheiden sich die Studien in der Reihenfolge und prozentualen Verteilung. Eine mögliche Erklärung für diese Abweichung kann mit den verschiedenen Herangehensweisen der Studien erklärt werden.

Im Folgenden sollen die ersten drei CMS mit den höchsten Marktanteilen der beiden Studien hinsichtlich der semantischen Funktionalität untersucht werden.

Zusätzlich soll das CMS *TYPO3* in Bezug auf die semantische Unterstützung betrachtet werden, obwohl es in den Analysen von Shreves und W3Techs zur Marktverbreitung jeweils auf dem sechsten Platz rangiert [25] [30]. Nach Angaben des TYPO3 Community Managers Van 't Ende liegt die Verbreitung bei mehr als 500.000 Installationen weltweit [28]. Ein Großteil entfällt dabei auf den europäischen, insbesondere den deutschsprachigen Raum.

[9] http://www.joomla.org
[10] http://wordpress.org
[11] http://drupal.org/

3.4.1 Joomla!

Joomla! ist nach Unstimmigkeiten um die Namensrechte aus dem CMS *Mambo* hervorgegangen. Die erste Version wurde im September 2005 angekündigt und kurz darauf der Öffentlichkeit zugänglich gemacht [15]. Joomla! erfüllt alle eingangs genannten Punkte eines CMS (s. Abschnitt 3.2), wobei einige Funktionalitäten durch Erweiterungen (Extensions) nachträglich hinzugefügt wurden. Durch diesen Mechanismus kann auch eine semantische Unterstützung im begrenzten Maße nachgerüstet werden.

Phuoc und Rakhmawati [22] stellen mit *JSyndication* eine Erweiterung vor, die auf Basis von *Triplify*[12] die Inhalte aus Joomla! in RDF/JSON-Format umwandelt. Diese können anschließend von einer semantischen Suchmaschine indiziert und weiter verarbeitet oder aggregiert und in einer Art Nachrichtenfeed dargestellt werden. Die Publikation stellt eine generische Lösung dar, die nach Angaben der Autoren auch für andere CMS wie Drupal oder WordPress denkbar sind. Eine Weiterentwicklung der Erweiterung findet nicht statt - ein Download ist ebenfalls nicht mehr möglich.

Der Dienst *Zemanta*[13] stellt eine weitere semantische Umsetzung für Joomla! zur Verfügung. Nach eigenen Angaben wird der unstrukturierte Text mit Hilfe von Natural Lanugage Processing (NLP), semantischen Algorithmen und statistischen Vergleichen auf seine Bedeutung analysiert.[14] Anschließend werden dem Redakteur Vorschläge unterbreitet, um den Inhalt mit Kategorien, Tags, Links oder Bildern in einem graphischen Editor aufzuwerten.

Die API für Entwickler unterstützt neben XML und JSON auch RDF/XML als Ausgabe der Daten. Damit die RDF-Triple im Sinne des Semantic Webs weiter verarbeitet werden können, existiert eine eigene Zemanta Ontologie[15], die Klassen und Eigenschaften der verschiedenen Vorschläge definiert.

Neben der Joomla! Erweiterung existieren unter anderem auch Umsetzungen für WordPress und Drupal. Die grundlegende Funktionsweise wurde bei der Portierung beibehalten.

3.4.2 WordPress

WordPress wurde als Blog-System konzipiert und erstmals im Jahr 2004 veröffentlicht. Bis zur zweiten Version im folgenden Jahr wurde WordPress um Plugins, statische Seiten und ein neues, erweitertes Rollen- bzw. Rechtema-

[12] http://triplify.org/

[13] http://www.zemanta.com/

[14] http://www.zemanta.com/faq/, Kopie auf CD-ROM (Datei onlinequellen/2011_zemanta_faq.pdf vom 13.09.2011).

[15] http://s.zemanta.com/ns

nagement erweitert.[16] Damit kann WordPress nach Kampffmeyers Definition (s. Abschnitt 3.2) dem Bereich der klassischen CMS zugeordnet werden.

Eine semantische Unterstützung wird ähnlich zu Joomla! durch die angesprochenen Plugins ermöglicht. Gegenüber Joomla! existieren bereits einige Ansätze, die aufgrund des Ursprungs von WordPress dem Bereich des Semantic Blogging zuzuordnen sind.

Möller et al. unterscheiden bezogen auf die semantische Implementierung zwischen struktur- und inhaltsbezogenen Metadaten [19]. Strukturbezogene Metadaten leiten sich aus den einzelnen Teilen des Blogs, wie Artikel, Kommentare oder Verweise auf andere Blogs ab. Dem gegenüber ergeben sich inhaltsbezogene Metadaten aus der Bedeutung bzw. dem Thema des eigentlichen Artikels.

Für die semantische Umsetzung verwenden die Autoren die *SIOC* (Semantically-Interlinked Online Communities) Ontologie, die ein Vokabular bereit stellt, um die Beziehungen zwischen Artikeln, Kommentaren und Autoren des Blogs zu modellieren. In der vorgeschlagenen Lösung wird die Umwandlung direkt von einem WordPress Plugin übernommen, um den Arbeitsaufwand für den Benutzer so minimal wie möglich zu halten.

Für die semantische Aufbereitung des Inhalts wird eine Verwendung von verschiedenen Ontologien vorgeschlagen, die je nach inhaltlicher Ausrichtung des Artikels variiert. Zum Beispiel wird ein Artikel der sich mit der Ankündigung für eine Veranstaltung befasst, die *iCalendar* Ontologie verwenden, wohingegen für eine Buchrezension die BibTeX Ontologie für die Aufbereitung von bibliografischen Metadaten in Frage kommt. Möller et al. nehmen an, dass die Daten bereits in einer maschinenlesbaren Form, z. B. als digitaler Kalender oder Bibliothek, auf dem Computer des Benutzers vorliegen [19]. Die Daten können dann mit Hilfe von öffentlichen oder eigenen APIs in RDF-Triples umgewandelt und anschließend in die WordPress Installation importiert werden. Der Import kann mit einer dedizierten Desktop-Anwendung für den Benutzer vereinfacht werden, was jedoch den Vorteil des Webs, die Plattformunabhängigkeit, aufhebt.

Eine weitere semantische Implementierung bietet das WordPress Plugin *Tagaroo*[17]. Es arbeitet vergleichbar zu Zemanta, greift jedoch auf *Calais*[18] zu. Der Web Service analysiert die unstrukturierten Texte hinsichtlich der Bedeutung und gibt inhaltlich passende Entitäten, Fakten oder Veranstaltungen zurück. Das Plugin Tagaroo verwendet die Daten anschließend um dem Redakteur eine Auswahl an Tags und Bildern vorzuschlagen, die den Artikel näher beschreiben und erweitern. Derzeit analysiert Calais nur Texte in den Sprachen Englisch, Französisch und Spanisch.[19]

[16] http://codex.wordpress.org/History, Kopie auf CD-ROM (Datei onlinequellen/2011_wordpress_history.pdf vom 13.09.2011).

[17] http://tagaroo.opencalais.com/

[18] http://www.opencalais.com/

[19] http://www.opencalais.com/genfaq, Kopie auf CD-ROM (Datei onlinequellen/2011_

Ähnlich zu Zemanta gibt es auch bei Calais die Möglichkeit die semantischen Metadaten als RDF-Graph zu verarbeiten. Dazu verfügt Calais über eine eigene Ontologie[20], welche das verwendete Wissen in Form von Klassen und Eigenschaften modelliert.

3.4.3 Drupal

Drupal wurde vom Erfinder Dries Buytaert anfänglich zum Austausch von Nachrichten mit Freunden konzipiert und unter dem Namen „Drop“ als Internetplattform betrieben. Erst im Jahr 2001 wurde die Software der Plattform unter dem Namen Drupal der Öffentlichkeit zur Verfügung gestellt.[21] Ein Blick in die Versionsgeschichte zeigt, dass Drupal mit der Version 4 im Jahr 2004 erste Grundlagen für ein flexibles CMS mit dem Schwerpunkt auf Community-Verwaltung legte. Eine größere Bekanntheit erlangte das Projekt mit den nächsten beiden Versionen und wurde fortan auch für größere, kommerzielle Internetseiten verwendet. Im Januar 2011 wurde die aktuelle Version 7 veröffentlicht.[22] Sie beinhaltet neben einer überarbeiteten Oberfläche für Administratoren und Redakteure auch eine Reihe an neuen APIs für Modul-Entwickler.

Durch ihre modulare Struktur kann eine Drupal-Installation flexibel an den gegebenen Anwendungsfall angepasst werden. Die Basis hierfür bilden die von der Community bereitgestellten Module. Als Besonderheit soll hervorgehoben werden, dass Module in Drupal aufeinander aufbauen können, wodurch spezielle Funktionen nur einmal implementiert und von weiteren Modulen wiederverwendet werden können [29].

Drupal verfügt, im Vergleich zu den anderen vorgestellten CMS, über eine Reihe von Modulen, die eine semantische Unterstützung ergänzt. Im folgenden soll daher eine Auswahl vorgestellt werden.

Eines der frühesten semantischen Module ist *SIOC*[23], das im Jahr 2007 für Drupal 4.7 veröffentlicht wurde. Ähnlich wie das von Möller et al. vorgeschlagene WordPress-Plugin, wird die SIOC Ontologie verwendet, um die Struktur einer Community-Seite mit Foren oder eines Blogs mit Kommentaren semantisch abzubilden und mit anderen Seiten zu verknüpfen. Ein Download des Moduls für Drupal 7 ist nicht erhältlich, da die Funktionalitäten in den Kern des CMS eingeflossen sind.[24]

---

opencalais_faq.pdf vom 13.09.2011).

[20] http://s.opencalais.com/

[21] http://drupal.org/about/history, Kopie auf CD-ROM (Datei onlinequellen/2011_drupal_history.pdf vom 13.09.2011).

[22] http://api.drupal.org/api/drupal/CHANGELOG.txt, Kopie auf CD-ROM (Datei onlinequellen/2011_drupal_changelog.pdf vom 13.09.2011).

[23] http://drupal.org/project/sioc

[24] http://drupal.org/node/743208, Kopie auf CD-ROM (Datei onlinequellen/2010_drupal_sioc-d7-update.pdf vom 13.09.2011).

Mit *Calais*[25] existiert ein weiteres semantisches Modul, das vergleichbar mit Tagaroo für WordPress ist. Es sendet die eingegebenen Texte an die Calais Web Service API und speichert die gefundenen Entitäten, Fakten oder Veranstaltungen ab.

Für die weitere Verarbeitung eröffnen sich vielfältige Szenarien, die über die Grenzen des Calais Moduls hinausgehen und von anderen Modulen übernommen werden können. Beispielsweise lassen sich Entitäten als Begriffe in der Drupal eigenen Taxonomie abspeichern und je nach Einstellung automatisch oder manuell vom Redakteur dem Artikel als Tags zuweisen.[26] Alternativ können aus dem Text extrahierte Orte auf einer Karte passend zum Inhalt visualisiert werden.

Ein zentraler Baustein von Drupal sind die Knoten (engl. Node), die bis auf ein paar Ausnahmen die Inhalte der Seite umfassen. Die Knoten folgen einer statischen Struktur aus Eingabefeldern, die wiederum in Inhaltstypen (engl. Content Types) definiert ist [29]. Standardmäßig existieren in einer Drupal-Installation zwei Inhaltstypen. Eigene Inhaltstypen können bis Drupal 7 nur durch das separat erhältliche Modul *Content Construction Kit*[27] (CCK) angelegt werden.

Corlosquet et al. stellen mit dem *RDF Content Construction Kit*[28] (RDF CCK) eine semantische Erweiterung des CCKs vor [10]. Mit dem Modul ist es möglich, den erstellten Inhaltstypen bestimmte Ontologie-Klassen und den zugehörigen Feldern die Ontologie-Eigenschaften zuzuweisen. Die Daten werden zum einen im RDF-Triple-Store abgelegt, damit beispielsweise SPARQL-Abfragen an einen Endpunkt gestellt werden können und zum anderen werden die Zuweisungen in Form von RDFa-Tags im HTML-Quellcode untergebracht. Zusätzlich existiert die Möglichkeit die semantischen Daten direkt als RDF/XML auszugeben.

Damit der Grundgedanke des Linked Data gewahrt wird, haben Corlosquet et al. zusätzlich den *RDF External Vocabulary Importer*[29] (Evoc) veröffentlicht [10]. Mit Hilfe des Moduls können fremde Ontologien eingelesen werden und stehen anschließend für die Zuweisungen im RDF CCK bereit.

Mit Drupal 7 wurde das CCK als integraler Bestandteil in Form der Field API ins CMS eingeführt. Auch SIOC, RDF CCK und weitere Module mit semantischer Unterstützung wurden zu weiten Teilen zusammengefasst und den Entwicklern als RDF API zur Verfügung gestellt.[30] Durch die Integration

[25] http://drupal.org/project/opencalais

[26] http://drupal.org/project/opencalais, Kopie auf CD-ROM (Datei onlinequellen/2011_drupal_opencalais.pdf vom 13.09.2011).

[27] http://drupal.org/project/cck

[28] http://drupal.org/project/rdfcck

[29] http://drupal.org/project/evoc

[30] http://drupal.org/project/rdfcck, Kopie auf CD-ROM (Datei onlinequellen/2011_drupal_rdfcck.pdf vom 13.09.2011).

in den CMS-Kern profitieren auch andere Module, die ihre Ausgabe damit einfacher mit semantischen Daten anreichern können.

Die Oberfläche zum Importieren von Ontologien und Zuweisen von Klassen und Eigenschaften wird nicht mit Drupal 7 ausgeliefert, sondern separat als Modul RDF UI weiterentwickelt und im Paket *RDF Extensions*[31] (RDFx) vertrieben.

Neologism ist ein RDFS-Vokabular Editor und ein weiteres Drupal-Projekt des Digital Enterprise Research Institute (DERI). Im Gegensatz zum RDF CCK Modul, das ebenfalls vom DERI entwickelt wurde, stellt Neologism eine Sammlung an Modulen bereit und nutzt Drupal daher als Framework für die Entwicklung.

Der Funktionsumfang von Neologism reicht vom Erstellen verschiedener Klassen, Eigenschaften und deren Beziehungen untereinander mit Hilfe einer visuellen Ansicht, über das Zuweisen von externen Vokabularen bis zum Ausliefern der Vokabulare mit Content Negotiation. Nach eigenen Angaben der Autoren erfüllt Neologism jedoch nicht den Anspruch und Umfang eines Ontologie-Editors.[32]

3.4.4 TYPO3

Kasper Skårhøj begann im Jahr 1997 mit den Arbeiten an einem CMS, welches erstmalig im Jahr 2000 als Beta-Version erhältlich war. Nach zwei Jahren wurde das CMS unter dem Namen *TYPO3*[33] in der finalen Version 3 offiziell der Öffentlichkeit zugänglich gemacht. Aktuell ist die Version 4.5 verfügbar, wobei parallel an der fünften Version gearbeitet wird.[34]

TYPO3 besitzt noch keine semantische Unterstützung. Erste Funktionalitäten in diese Richtung sollen erst mit der Version 4.7 in das CMS einfließen.[35] Für die Entwicklung wurde die Arbeitsgruppe *Semantic Category, Ontology, Tag and Taxonomy System* (S.C.O.T.T.Y.) ins Leben gerufen, die sich die Integration einer Taxonomie und des Semantic Webs zum Ziel gesetzt hat. Letzteres wurde bereits als separate Erweiterung im *TYPO3 Extension Repository*[36] veröffentlicht und kann mit der aktuellen TYPO3 Version eingesetzt werden. Die Extension bietet dem Redakteur ein neues Inhaltselement an, welches mit einer SPARQL-Query konfiguriert wird und die empfangenen Ergebnisse anschließend formatiert auf der Seite ausgibt. In den nächsten Schritten sollen weitere Komponenten zum Produzieren und

[31] http://drupal.org/project/rdfx

[32] http://neologism.deri.ie/about, Kopie auf CD-ROM (Datei onlinequellen/2011_neologism_about.pdf vom 13.09.2011).

[33] http://typo3.org/

[34] http://typo3.com/History.1268.0.html, Kopie auf CD-ROM (Datei onlinequellen/2011_typo3_history.pdf vom 13.09.2011).

[35] http://forge.typo3.org/projects/extension-scotty, Kopie auf CD-ROM (Datei onlinequellen/2011_typo3_scotty.pdf vom 13.09.2011).

[36] http://typo3.org/extensions/repository/view/semantic/current/

Konsumieren der Linked Data Cloud folgen.[37]

Wahl et al. [31] haben sich im akademischen Bereich mit einer semantischen Unterstützung für TYPO3 auseinander gesetzt und schlagen eine Lösung zum Erstellen von RDF angereicherten Produkt-Katalogen vor. Der *RDFCreator* setzt eine Ontologie im OWL-Format voraus, die vom System importiert und geparst wird. Redakteure können, basierend auf den Regeln der Ontologie, Produkte zum Katalog hinzufügen. Abschließend kann der Katalog als RDF-Datei exportiert und weiterverarbeitet werden. Eine Integration ist nach Angaben der Autoren auch für andere PHP basierte CMS möglich [31]. Warum der RDFCreator für TYPO3 implementiert wurde und wie die weitere Verzahnung mit dem CMS aussieht, lässt die Publikation allerdings offen.

[37] http://forge.typo3.org/projects/extension-semantic/wiki/, Kopie auf CD-ROM (Datei onlinequellen/2011_typo3_semantic-web.pdf vom 13.09.2011).

Kapitel 4

Ontologie-Entwicklung

Der Begriff Ontologie-Entwicklung leitet sich direkt aus dem englischen Begriff „Ontology Engineering“ ab und beschäftigt sich mit dem Ausdruck spezieller Wissensgebiete in Form von Ontologien [23]. Die Umsetzung kann in verschiedenen Beschreibungssprachen geschehen und umfasst mehrere Arbeitsschritte. Diese sollen im ersten Abschnitt mit dem manuellen Ontologie-Entwicklungszyklus näher erläutert werden. Im folgenden Abschnitt wird die Idee einer Website-Ontologie formuliert und deren Besonderheiten ausgeführt.

4.1 Manuelle Ontologie-Entwicklung

Noy und McGuinness stellen in ihrer Publikation eine Reihe von Schritten zum manuellen Erstellen einer Ontologie auf. Die Abbildung 4.1 wurde auf Basis der Arbeit erstellt und visualisiert die einzelnen Schritte [20]. Da die Erstellung ein iterativer Prozess ist, kann die Abfolge sowohl als gesamter Zyklus durchlaufen, als auch einzelne Schritte variabel ausgelassen oder wiederholt werden. Die nachfolgenden Erläuterungen beziehen sich auf die angesprochene Publikation von Noy und McGuinness [20].

Im Ausgangspunkt des Zyklus wird der Bereich der Ontologie festgelegt. Eine Ontologie wird für einen bestimmten Zweck modelliert und deckt ein abgegrenztes Wissensgebiet ab. Der Bereich der Ontologie sollte lediglich so groß gewählt werden, dass er den Verwendungszweck, z. B. den Einsatz in der Softwareentwicklung oder in Prozessabläufen, erfüllt. Des Weiteren hilft es in diesem Schritt festzulegen, welche Fragen die Ontologie später durch automatisches Schlussfolgern beantworten wird. Dadurch ergibt sich unter Umständen bereits ein spezieller Blickwinkel auf das Wissensgebiet, das die Ontologie erfüllen wird. Folglich existiert keine einzig „richtige“ Ontologie für ein Gebiet. Jede Ontologie bringt ein spezifisches Wissen für ein Gebiet ein.

Sind die grundlegenden Anforderungen der Ontologie geklärt, sollte vor der eigentlichen Modellierung eruiert werden, ob bereits eine bestehende

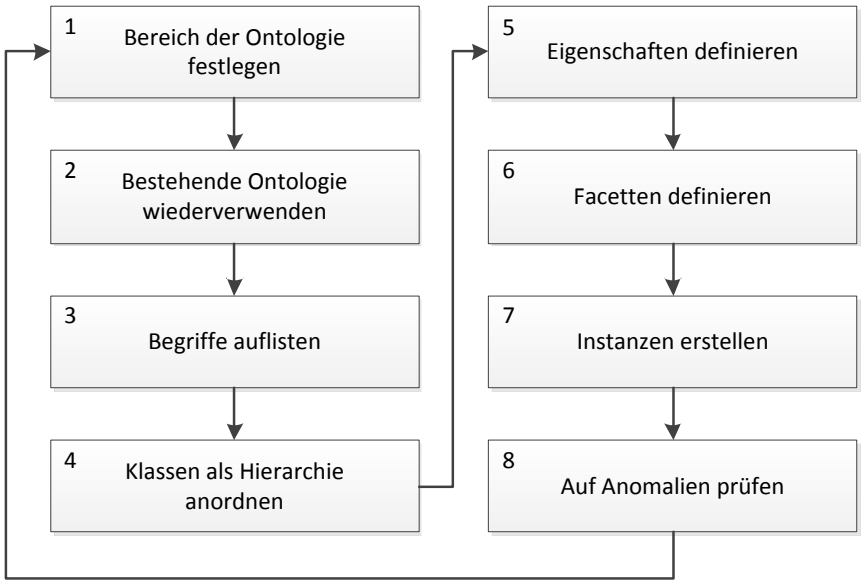

Abbildung 4.1: Manueller Entwicklungszyklus einer Ontologie nach Noy und McGuinness [20].

Ontologie für den geforderten Zweck existiert. Für die Recherche können Ontologie-Suchmaschinen, wie *Swoogle*[1], *Watson*[2] oder *OntoSearch*[3], sowie Verzeichnisse, wie *Protégé Ontology Library* [4], herangezogen werden. Existiert eine passende Ontologie, so ist die Wiederverwendung der eigenen Entwicklung vorzuziehen. Erfüllt die existierende Ontologie die Anforderungen nur teilweise, so sollte eine Verknüpfung zwischen den Klassen und Eigenschaften (siehe Abschnitt 2.2.3) gesetzt werden. Wird keine passende Ontologie gefunden, so kann eine eigene durch die nachfolgenden Schritte modelliert werden.

Im dritten Schritt der Modellierung werden markante Begriffe des Bereiches der Ontologie als unstrukturierte Liste festgehalten. Es empfiehlt sich, Substantive und Verben getrennt voneinander zu notieren, da diese in weiterer Folge für die Definition von Klassen und Eigenschaften verwendet werden.

Nachdem die relevanten Begriffe identifiziert sind, werden in Schritt vier die Klassen in einer Hierarchie angeordnet. Als Vorgehensweise werden von den Autoren die Top-Down oder Bottom-Up, sowie eine Mischform vorgeschlagen [20]. Bei der Top-Down-Vorgehensweise, werden zuerst die allgemeinen Begriffe festgelegt, die nach und nach durch weitere Begriffe spezifiziert werden. Die Bottom-Up-Methode beschreibt den umgekehrten Anwendungs-

[1] http://swoogle.umbc.edu/
[2] http://watson.kmi.open.ac.uk/WatsonWUI/
[3] http://www.ontosearch.org/
[4] http://protegewiki.stanford.edu/wiki/Protege_Ontology_Library

fall, der von einem spezifischen Begriff ausgeht und weitere generalisierte Begriffe zuordnet. In den häufigsten Fällen verwendet man jedoch eine Mischform aus beiden Methoden. Bei der Konstruktion der Hierarchie ist folgendes zu beachten: Wenn eine Subklasse A von einer Klasse B abgeleitet ist, dann ist jede Instanz von A gleichzeitig eine Instanz von B. Das ist deshalb von entscheidender Bedeutung, da es andernfalls zu einer inkonsistenten Ontologie kommen kann [1].

Im fünften Schritt werden die identifizierten Eigenschaften den Klassen zugeordnet. Die neuen Eigenschaften können intrinsischer (z. B. Farbe, Geschmack, usw.) oder extrinsischer (z. B. Name, Status, usw.) Natur sein. Darüber hinaus können die Eigenschaften Beziehungen zwischen Klassen oder Teile der Klasse beschreiben. Generell wird für diesen Arbeitsschritt empfohlen, die Eigenschaften möglichst weit an die Spitze der Hierarchie den passenden Klassen zu zuweisen, damit die Eigenschaften auch in den abgeleiteten Klassen verwendet werden können [20].

Anschließend werden für die Eigenschaften verschiedene Facetten definiert. Bei einer Facette kann es sich z. B. um die Einschränkung der zugelassenen und verwendeten Werte oder die Anzahl der Werte handeln. Für die Modellierung ist die gesamte Bandbreite von OWL zur detaillierten Beschreibung der Eigenschaften verfügbar. Das heißt im Detail, dass die vorgegebenen Werte für eine Eigenschaft mit `owl:hasValue`, `owl:someValuesFrom`, usw. als zulässige Auswahl für die Instanzen definiert werden. Darüber hinaus lässt sich die Eigenschaft auf einen bestimmten Datentyp, wie String, Number oder Boolean, sowie Klassen einschränken. Letzteres geht mit der Verwendung von `rdfs:domain` und `rdfs:range` einher (s. Abschnitt 2.2.2). Die Kardinalität kann über die bekannten Elemente `owl:cardinality`, `owl:minCardinality` und `owl:maxCardinality` (s. Abschnitt 2.2.3) justiert werden. Des Weiteren kann die Eigenschaft als symmetrisch, transitiv, funktional, usw. deklariert werden.

Mit den definierten Klassen und Eigenschaften lassen sich im nächsten Arbeitsschritt konkrete Instanzen erstellen. Durch die erstellte Struktur und die Instanzen können automatische Schlussfolgerungen zwischen den Instanzen gezogen werden.

Antoniou et al. fügen der Abfolge noch einen weiteren Arbeitsschritt (Schritt acht) hinzu, in dem die Ontologie auf Anomalien überprüft wird [1]. Das schließt insbesondere eine Analyse der internen Konsistenz ein. Die häufigsten Fehlerquellen sind eine falsche Kardinalität oder unzulässige Einschränkung des Wertebereichs mit `rdfs:domain` und `rdfs:range`.

Nachdem der letzte Arbeitsschritt vollzogen wurde, kann die Ontologie, ähnlich einer Software, den Anforderungen gegenübergestellt werden, woraus sich unter Umständen eine Modifikation der Ontologie ergibt und der Zyklus erneut durchlaufen wird.

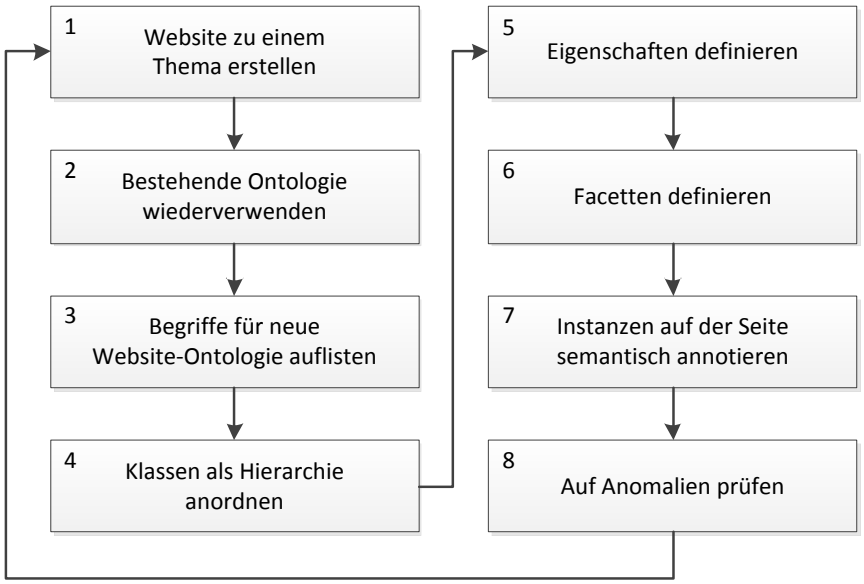

Abbildung 4.2: Entwicklungszyklus einer Website-Ontologie.

4.2 Entwicklungszyklus von Website-Ontologien

Wie bereits in Abschnitt 2.2.3 ausgeführt, werden Ontologien in der Informatik benutzt, um Wissen in maschinenverständlicher Form darzustellen. Dabei deckt eine Ontologie in den meisten Fällen nur einen definierten Teilbereich eines Themas oder Wissensgebietes ab. Sie wird wie im vorangehenden Abschnitt häufig auf dem manuellen Weg erstellt.

Die Abbildung 4.2 stellt den Entwicklungszyklus einer Website-Ontologie dar, welche auf der Abbildung 4.1 von Noy und McGuinness [20] basiert.

Die Grundannahme für die Entwicklung einer Website-Ontologie ist, dass eine Website Informationen zu einem konkreten Thema publiziert. Folglich kann angenommen werden, dass der Anbieter über Domänenwissen verfügt und ein Spezialist auf dem Themengebiet ist. Um eine Wiederverwendung des Wissens im Sinne des Semantic Webs zu ermöglichen, sollte dieses Wissen in Form von Beziehungen und Fakten als Ontologie aufbereitet und abgelegt werden.

An dieser Stelle sei angemerkt, dass sich Website-Ontologien nicht für jede Art von Internetangebot eignen. Beispielsweise lassen sich die Ontologien nur schwer für Portale mit einer Vielzahl an Themengebieten umsetzen. Als Lösung für das Problem könnten mehrere differenzierte Ontologien erstellt und veröffentlicht werden.

Für Website-Ontologien gilt, wie schon für die manuelle Entwicklung, dass im zweiten Schritt geprüft werden sollte, ob bereits thematisch passende Ontologien existieren. Diese Ontologien sind der eigenen Entwicklung

vorzuziehen. Andernfalls kann im nächsten Schritt mit der eigentlichen Erstellung fortgefahren werden.

Darauffolgend sollten wichtige Begriffe aus der Website extrahiert und getrennt nach Substantiven und Verben vermerkt werden. Diese werden anschließend zu Klassen und Eigenschaften der Ontologie verarbeitet. Die Schritte vier bis sechs für die Erstellung von Klassen und Eigenschaften entsprechen denen im Abschnitt 4.1 und sollen an dieser Stelle nicht weiter beschrieben werden.

Der einzige Schritt, der sich im weiteren Ablauf noch vom manuellen Weg unterscheidet, ist das Anlegen von Instanzen (siebenter Schritt). Für Website-Ontologien bietet es sich an, die Instanzen direkt auf der Website zu integrieren und bestimmte Bereiche bzw. Begriffe semantisch zu hinterlegen.

Das Prüfen auf Anomalien (achter Schritt) in der Ontologie erfolgt wieder analog zur manuellen Ontologie-Entwicklung.

Kapitel 5

Praxisteil

Dieses Kapitel widmet sich der Implementierung des Prototyps, welcher auf Basis der Erkenntnisse aus Kapitel 4 umgesetzt wurde. Der Entwicklungszyklus einer Website-Ontologie (s. Abschnitt 4.2) soll im folgenden Kapitel teilautomatisiert durch ein CMS geschehen und so eine Arbeitserleichterung schaffen. Dazu werden im ersten Abschnitt die Anforderungen an die Umsetzung formuliert und nachfolgend markante Aspekte der Implementierung besprochen. Abschließend wird auf Probleme bei der Umsetzung eingegangen.

5.1 Anforderungen

Das Ziel des Prototyps ist es, eine Website-Ontologie zu erstellen und diese im RDF/XML-Format für die weitere Verwendung auszugeben. Um die Entwicklung zu vereinfachen, wird auf ein CMS zurückgegriffen.

Im Folgenden werden die einzelnen Schritte des Entwicklungszyklus einer Website-Ontologie (s. Abb. 4.2) dahingehend betrachtet, ein passendes CMS für die Implementierung auszuwählen und die einzelnen Anforderungen für den Prototypen abzuleiten.

Dabei ist zu beachten, dass die Wahl des CMS die Rahmenbedingungen für den Prototyp festlegt und die Umsetzung je nach Entscheidung variieren wird. Im Allgemeinen sollte das System möglichst flexibel agieren und offen für externe Erweiterungen sein, sodass keine direkten Modifikationen am Kern des CMS notwendig sind. Wie bereits im Rahmen dieser Arbeit ausgeführt, existiert in allen untersuchten CMS eine Schnittstelle für Module oder Erweiterungen. Folglich kann keines der aufgeführten Systeme von vornherein für eine Umsetzung ausgeschlossen werden.

Der erste Schritt der Abbildung umfasst die Erstellung der eigentlichen Website, die vom Benutzer z. B. mit einem der vorgestellten CM-Systeme (s. Abschnitt 3.4) umgesetzt werden kann. Betrachtet man den Punkt unabhängig von der Entwicklung einer Website-Ontologie, so unterscheiden sich die

Systeme im Funktionsumfang sowie der zugrunde liegenden Herangehensweise und Verwaltung der Inhalte. Folglich müssen für eine Entscheidung weitere Parameter herangezogen werden.

Damit im zweiten Schritt eine bestehende Ontologie wiederverwendet werden kann, muss der Benutzer eine manuelle Recherche durchführen. Dieser Schritt kann nicht vom CMS durchgeführt werden, da das System nicht in der Lage ist, die Ontologien hinsichtlich der zukünftigen Verwendung zu bewerten. Jedoch sollte es den Benutzer nach der Auswahl bei der Wiederverwendung unterstützen. Drupal löst diese Anforderung durch das Modul *RDF External Vocabulary Importer*[1] (Evoc), welches bestehende Ontologien importiert und diese für die weitere Nutzung im CMS aufbereitet. In anderen CMS kann eine Ontologie nur manuell, durch das Einbinden der entsprechenden Ontologie in der Seitenausgabe wiederverwendet werden. Eine systemseitige Unterstützung ist nicht gegeben.

Hat sich der Benutzer mit dem dritten Schritt dazu entschieden, eine neue Website-Ontologie anzulegen, ist die Anforderung an das CMS, dem Benutzer bei der Auswahl der wichtigsten Begriffe zu assistieren. Das kann zum einen durch die Analyse des Website-Inhaltes mittels Calais oder Zemanta (s. Abschnitt 3.4) oder zum anderen durch die Auflistung der Navigationspunkte erfolgen. Da die Menge an gefundenen Begriffen unter Umständen zu gering oder für eine Ontologie nicht relevant ist, sollte der Benutzer anschließend weitere Begriffe finden und diese zur Liste hinzufügen.

Die nächsten Schritte können nur durch dedizierte Benutzeraktionen abgeschlossen werden. Das umfasst die Anordnung der Klassen zu einer Hierarchie sowie die Definition von Eigenschaften und deren zugehörigen Facetten. Einzig das von Corlosquet et al. vorgestellte Modul *RDF Content Construction Kit*[2] (RDF CCK) ermöglicht die Bildung eines semantischen Vokabulars aus Klassen und Eigenschaften [10]. Eine detaillierte Erläuterung der Funktionsweise befindet sich im Abschnitt 5.2.

Die Instanziierung der Klassen und Eigenschaften im siebenten Schritt sollte ebenfalls teilautomatisiert vom CMS durchgeführt werden. Dazu verwendet der Benutzer entweder die erstellten Klassen und Eigenschaften direkt in der Seitenausgabe (ähnlich dem zweiten Schritt) oder das CMS übernimmt die semantische Annotation.

Die Überprüfung auf Anomalien im achten Schritt kann aufgrund der Komplexität des logischen Schlussfolgerns nicht vom CMS durchgeführt werden. Stattdessen muss die Ontologie durch eine externe Anwendungen kontrolliert werden.

Zusammenfassend lässt sich sagen, dass sich alle untersuchten CMS um eigene Implementierungen erweitern lassen. Drupal bietet jedoch durch die bestehenden Module für ein semantisches Vokabular die besten Vorausset-

[1] http://drupal.org/project/evoc
[2] http://drupal.org/project/rdfcck

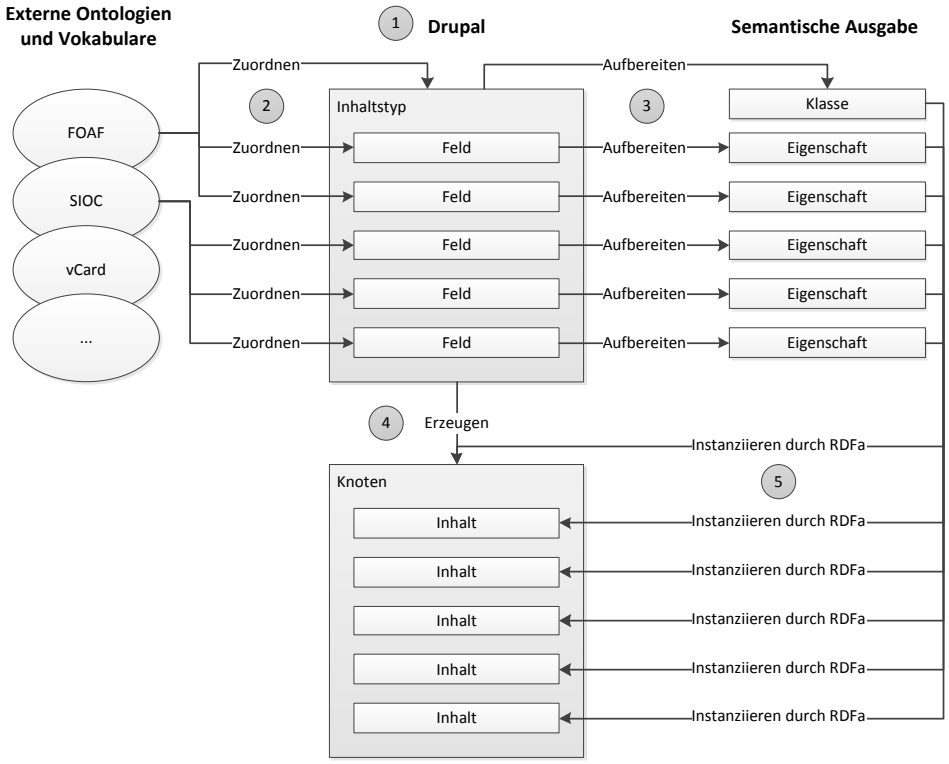

Abbildung 5.1: Semantische Zuordnung von externen Vokabularen und Ontologien sowie die semantische Aufbereitung von Drupal-Inhaltstypen nach Corlosquet et al. [10].

zungen für die Erstellung einer Website-Ontologie. Daher soll im Folgenden auf die bestehende Arbeit von Corlosquet et al. aufgebaut und eine eigene Implementierung entworfen werden.

5.2 Ansatz

Zu Beginn dieses Abschnitts soll ein Überblick über die Funktionsweise der Drupal-Module von Corlosquet et al. gegeben werden. Das Verständnis ist für die weitere Arbeit von Vorteil, da dieser Ansatz als Grundlage für die weitere Implementierung dient. Anschließend soll der vom Autor gewählte Ansatz genauer erläutert werden.

5.2.1 Bestehender Ansatz

Corlosquet et al. stellen in ihrer Arbeit [10] eine Reihe von Drupal-Modulen vor, die eine semantische Aufbereitung der Inhalte ermöglichen. Die Funktionsweise ist in Abbildung 5.1 dargestellt und basiert auf der Publikation.

Der erste Punkt illustriert den Aufbau der Inhaltstypen (engl. Content Types). Ein Inhaltstyp in Drupal umfasst eine Ansammlung von Formularfeldern, wobei mindestens ein Textfeld pro Typ als Titel vorhanden sein muss. Weitere Felder können je nach Anwendungsfall hinzugefügt werden und bestehen z. B. aus einfachem Text, Auswahllisten, Links usw. Die Felder können mit bestimmten Werten für die Redakteure vorbelegt oder eingegrenzt werden. Bei einem Inhaltstyp kann es sich z. B. um einen Kontakt im Adressbuch handeln, wobei Name, Adresse, Telefonnummer, usw. als Felder definiert sind. Inhaltstypen werden somit als Vorlage für die Redakteure verwendet, welche sich bei der späteren Arbeit lediglich um die Eingabe der Inhalte, nicht jedoch um die Struktur, sorgen müssen.

In der vorgestellten Publikation wird der zweite Punkt der Abbildung von zwei Modulen übernommen. Das Modul Evoc importiert externe Vokabulare und Ontologien in Drupal und übergibt diese an das RDF CCK. Das letztgenannte Modul stellt eine Oberfläche bereit, um die semantischen Klassen und Eigenschaften dem Inhaltstyp respektive den Feldern zuzuordnen. Seit Drupal 7 wurde ein Teil des RDF CCK, welches für Drupal 6 entwickelt wurde, in den Kern übernommen und die Oberfläche als zusätzliches Modul im Paket RDF Extensions (RDFx) veröffentlicht.

Eine Aufbereitung der Inhaltstypen in Form eines Site-Vokabulars im dritten Punkt wurde ebenfalls von Corlosquet et al. vorgeschlagen und zum Teil auch im RDF CKK für Drupal 6 implementiert [11]. Es handelt sich dabei um den inversen Fall der Zuweisung aus dem zweiten Punkt. Folglich werden die Inhaltstypen als RDFS-Klassen und die Felder als RDFS-Eigenschaften ausgegeben. Existiert eine Zuweisung von einem importierten Vokabular, so wird das neue Element mit `rdfs:subClassOf` bzw. `rdfs:subPropertyOf` von dem Original-Element abgeleitet. Alle neuen Elemente werden im Namensraum der Website abgelegt und ergeben somit ein Site-Vokabular. In der aktuellen Implementierung für Drupal 7 wurde die Funktionalität bis zum Zeitpunkt dieser Arbeit nicht übernommen. Damit ist die Erstellung eines Site-Vokabulars derzeit nicht möglich.

Im vierten Punkt werden die abstrakten Inhaltstypen vom Redakteur mit Inhalt gefüllt und in Form eines neuen Knoten (engl. Node) in Drupal abgelegt. Der Vorgang kann vom Redakteur beliebig oft wiederholt werden. Da Drupal knotenbasiert arbeitet, stellt jeder Knoten gleichzeitig eine Seite dar und kann (je nach Freigabe-Einstellungen) von Besuchern der Seite aufgerufen werden.

Die Knoten werden für die Ausgabe in Drupal 6 vom RDF CCK und ab Drupal 7 standardmäßig mit RDFa annotiert. RDFa erweitert den HTML-Standard dahingehend, dass die Elemente eines Vokabulars oder einer Ontologie, in diesem speziellen Fall des Site-Vokabulars (nur Drupal 6) bzw. des importierten Vokabulars, direkt mit den HTML-Attributen in den Quelltext der Seite geschrieben werden. Der Vorgang entspricht damit dem Instanzieren von Elementen eines Vokabulars respektive einer Ontologie und kann

ebenfalls von Maschinen gelesen und weiterverarbeitet werden.

5.2.2 Eigener Ansatz

Die Idee eines Site-Vokabulars, welche von Corlosquet et al. [11] publiziert wurde, besitzt einige entscheidende Nachteile. So werden die externen Elemente als Eltern-Verweise für die neuen Site-Vokabular-Elemente und letztere dann als RDFa im HTML-Quelltext verwendet. Durch diese zusätzliche Ebene wird die semantische Struktur komplexer, ohne einen Mehrwert zu schaffen, da keine zusätzlichen Informationen zu den neuen Elementen hinzugefügt werden. Die bessere Alternative wäre daher, die Elemente aus den externen Vokabularen und Ontologien direkt den Inhaltstypen zuzuordnen und nicht den Umweg über das Site-Vokabular zu nehmen.

Des Weiteren wurde festgestellt, dass die Anzahl der Elemente im Vokabular immer äquivalent zur Anzahl der angelegten Inhaltstypen in Drupal ist. Das ist insofern ein gelungener Zug, da zu jedem definierten Element mindestens eine Instanz in Form eines Knoten vorhanden ist. Der Nachteil ist jedoch, dass sich mit der geringen Anzahl an Inhaltstypen keine umfangreichen Beziehungen zwischen den verschiedenen Elementen modellieren lassen.

Für den eigenen Ansatz wurde die grundsätzliche Idee des Site-Vokabulars übernommen und zur Website-Ontologie weiterentwickelt. Die Abbildung 5.2 veranschaulicht, dass die Inhaltstypen weiterhin genutzt werden, um diese in Klassen und Eigenschaften auszudrücken. Zusätzlich kann der Benutzer weitere Elemente manuell hinzufügen, womit der dritte Punkt aus dem Entwicklungszyklus der Website-Ontologie (s. Abb. 4.2) erfüllt wird und das CMS dem Benutzer bei der Suche nach möglichen Begriffen für die Ontologie assistiert.

Des Weiteren können alle Elemente, gleich ob aus Inhaltstypen oder manuell vom Benutzer erzeugt, untereinander in Beziehung gesetzt werden. Das geschieht indem die Elemente in einer Hierarchie angeordnet oder individuelle Wertebereiche für die Eigenschaften festgelegt werden. Zusätzliche Anweisungen in Form von Kardinalitäten, Arten von Beziehungen, Regeln, usw. sind ebenfalls denkbar.

Erst durch diese beiden Erweiterungen kann die Idee einer Website-Ontologie realisiert werden. Dabei ist jedoch zu beachten, dass das Modul keinen Ersatz zu einem professionellen Ontologie-Editor darstellen wird. Vielmehr soll es den Benutzer bei der semantischen Wiederverwendung und Aufbereitung der Seite unterstützen und eine Lösung in der gewohnten Arbeitsumgebung anbieten.

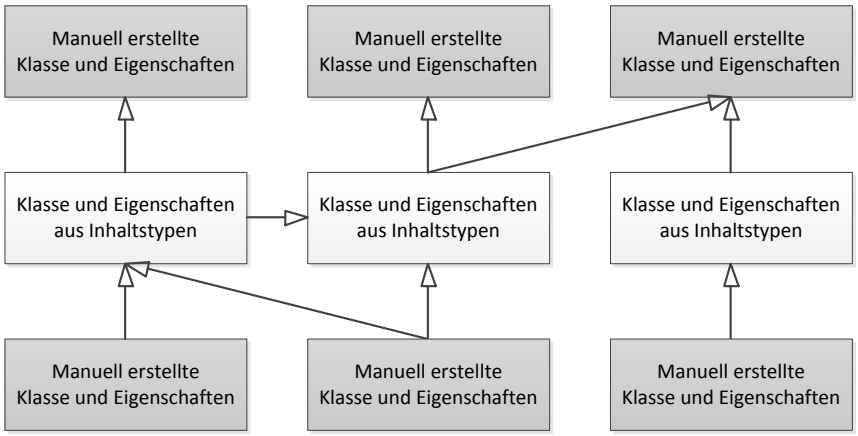

Abbildung 5.2: Erweiterung der Klassen und Eigenschaften aus Inhaltstypen um manuelle Elemente, welche unabhängig der Herkunft in Beziehung gesetzt werden können und so z. B. eine Hierarchie bilden.

5.3 Funktionsweise

Bevor im Einzelnen auf die Implementierung eingegangen wird, beschreibt dieser Abschnitt die Funktionsweise des entwickelten Drupal-Moduls. Für die Beschreibung wird das Szenario (s. Abschnitt 5.5) herangezogen.

Installation des Moduls

Ein Drupal-Modul wird über den Dialog im Bereich Administration » Module installiert. Der Benutzer wählt dazu ein komprimiertes Archiv von einem entfernten Server oder dem eignen Computer aus und startet die Installation. Nach Abschluss des Vorgangs wird das neue Modul in der Liste, die sich in verschiedene Bereiche gliedert, aufgeführt. Falls nicht bei der Installation geschehen, kann das Modul an dieser Stelle über die Auswahl aktiviert werden. Die Abbildung 5.3 stellt das installierte und aktivierte Website-Ontologie-Modul dar. Anschließend steht das Modul innerhalb von Drupal zur Verfügung.

Nach der Installation bietet das Modul zudem die Möglichkeit diverse Rechte (engl. Permissions) der Drupal-Benutzer einzuschränken. Dazu zählen u. a. die Zugriffsrechte auf den neuen Bereich oder die Möglichkeit eine Klasse bzw. Eigenschaft anzulegen, zu editieren oder zu löschen.

Transformation von Inhaltstypen

Wie bereits im Abschnitt Ansatz (s. Abschnitt 5.2) beschrieben, basiert das implementierte Modul auf den Inhaltstypen von Drupal. Diese können im CMS Administrationsbereich unter Struktur » Inhaltstypen verwaltet wer-

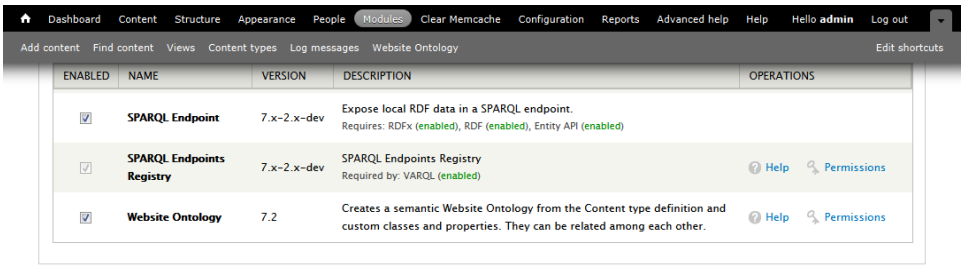

Abbildung 5.3: Aktivieren des Moduls über Administration » Module » RDF » Website Ontology.

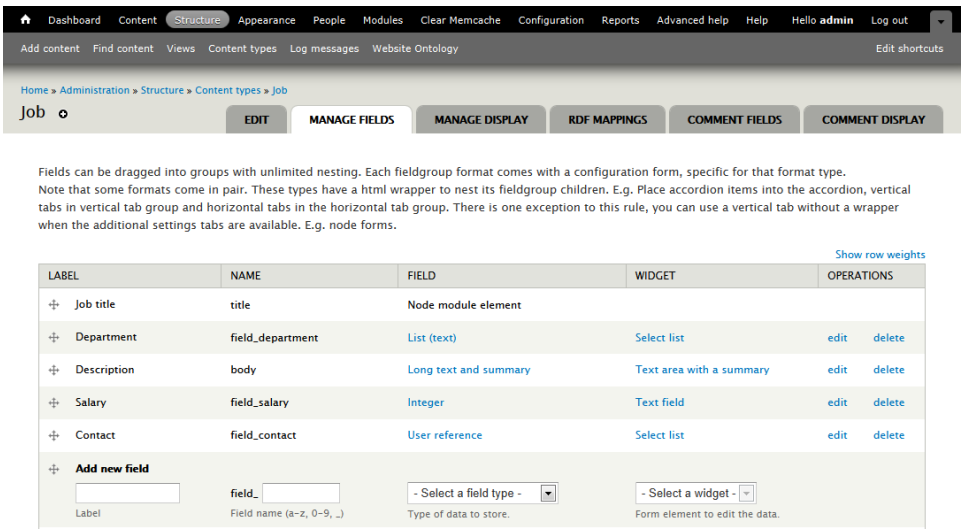

Abbildung 5.4: Dialog zum Erstellen eines neuen Inhaltstypen mit dessen zugehörigen Feldern. Änderungen werden automatisch erkannt und fließen in die Website-Ontologie ein.

den. Die Abbildung 5.4 stellt die Felder für den Inhaltstyp „Job“ dar. Je nach Anwendungsfall können die Inhaltstypen mit einer beliebigen Anzahl und Art von Feldern konfiguriert werden.

Erstellt der Benutzer einen neuen Inhaltstyp, wird dieser im Hintergrund vom Modul automatisch als äquivalente Klasse in der Website-Ontologie hinzugefügt. Felder werden ebenfalls automatisch in Eigenschaften transformiert und mit der vorher erstellten Klasse verknüpft.

Aktionen wie das Editieren oder Löschen eines Inhaltstyps bzw. Feldes werden ebenfalls vom Modul erkannt und automatisch auf die Website-Ontologie angewandt.

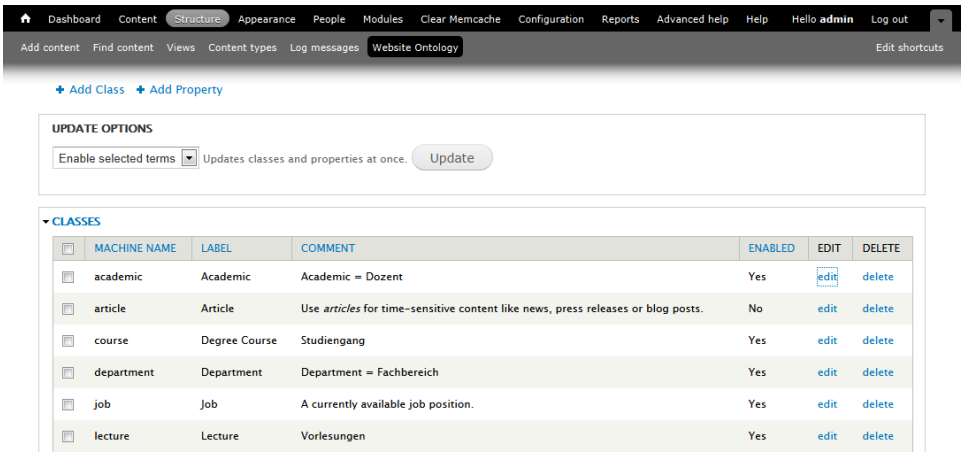

Abbildung 5.5: Die Listenansicht stellt Klassen und Eigenschaften der Website-Ontologie dar.

Verwaltung der Website-Ontologie

Die Abbildung 5.5 stellt die Verwaltung der Website-Ontologie dar, welche im Administrationsbereich über den Menüpunkt Struktur aufgerufen wird. Die Oberfläche der Listenansicht gliedert sich in vier wesentliche Bereiche. Am oberen Rand befinden sich die Links zum Anlegen einer neuen Klasse bzw. Eigenschaft. Das anschließende Formularfeld ermöglicht mehrere ausgewählte Klassen und Eigenschaften mit einem Mal zu aktivieren oder zu deaktivieren. Darunter befinden sich zwei Listen mit Klassen bzw. Eigenschaften der Website-Ontologie. Zu jedem Listeneintrag wird der eindeutige Maschinenname, das Label sowie der Kommentar angezeigt. Zusätzlich gibt eine eigene Spalte an, ob der Begriff zurzeit in der Website-Ontologie verfügbar ist. Eine gesonderte Auszeichnung von transformierten Inhaltstypen ist derzeit nicht vorgesehen, könnte in Zukunft jedoch den Umgang mit den Elementen vereinfachen.

Verarbeitung von Begriffen

Legt der Benutzer einen neuen Begriff an oder selektiert er einen Begriff zum Bearbeiten aus der Listenansicht, wechselt die Ansicht in den Editiermodus, welcher in Abbildung 5.6 dargestellt ist. Das Formular beinhaltet das Label und den Maschinennamen als Pflichtfeld. Der Maschinenname wird dabei automatisch von Drupal aus dem Label generiert, kann jedoch vom Benutzer manuell angepasst werden. Das Label und der Kommentar fördern das (menschliche) Verständnis beim Lesen der Begriffe in der späteren XML-Ausgabe. Damit diese Angabe sinnvoll verarbeitet werden kann, sollte die Sprache angegeben werden. Standardmäßig entspricht die Auswahl im For-

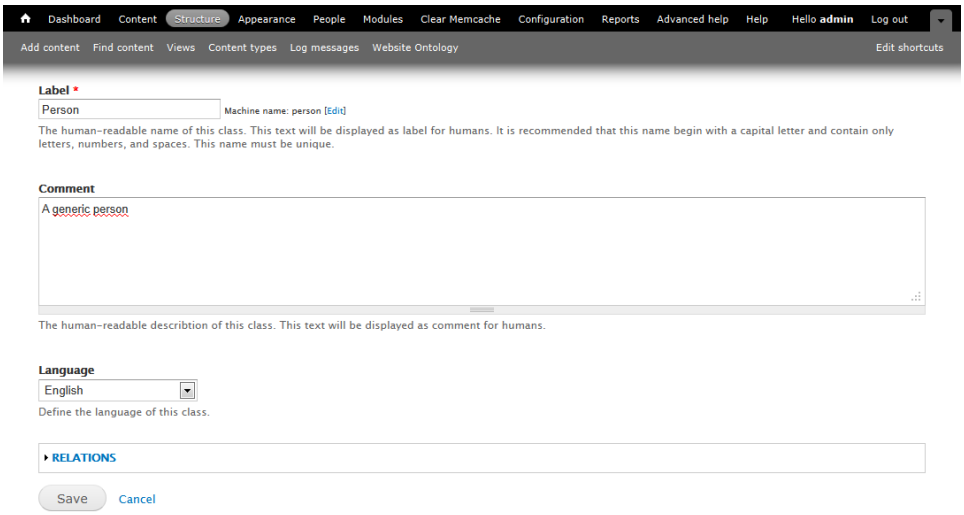

Abbildung 5.6: Das Formular zum Editieren von Klassen mit der Besonderheit, dass der Maschinenname automatisch aus dem Label ermittelt wird.

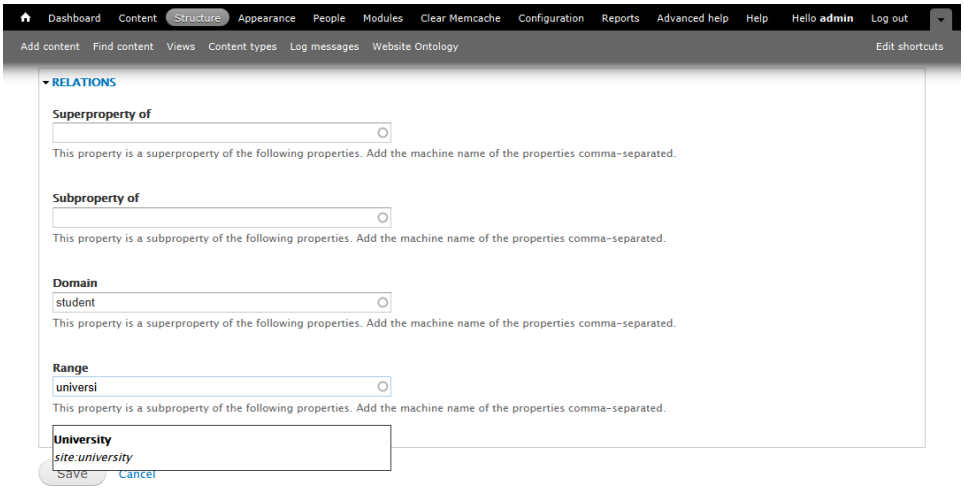

Abbildung 5.7: Formular zum Verknüpfen von Begriffen in Form einer Hierarchie und Wertebereich (nur bei Eigenschaften). Die automatische Vervollständigung assistiert dem Benutzer in allen Feldern.

mularfeld der Einstellung im Benutzerprofil von Drupal, kann jedoch frei angepasst werden.

Der Abschnitt zum Verknüpfen (engl. Relations) von Eigenschaften ist in Abbildung 5.7 dargestellt. Die Begriffshierarchie wird über die Felder „Superproperty of“ und „Subproperty of“ (respektive „Superclass of“ und „Subclass of“ für Klassen) gebildet. Das Feld „Superproperty of“ gibt an, dass

```
1 <rdf:RDF xmlns:rdf="&rdf;" xmlns:rdfs="&rdfs;" xmlns:owl="&owl;" xmlns:
      xsd="&xsd;" xmlns:site="&site;">
2 <owl:Ontology rdf:about="&site;"><rdfs:label>Website Ontology</rdfs:
      label></owl:Ontology>
3 <owl:Class rdf:about="&site;professor">
4   <rdfs:isDefinedBy rdf:resource="&site;"/>
5   <rdfs:subClassOf rdf:resource="&site;academic"/>
6   <rdfs:label xml:lang="en"><![CDATA[Professor]]></rdfs:label>
7   <rdfs:comment xml:lang="en"><![CDATA[Professor teaches students and
      works at the university]]></rdfs:comment>
8 </owl:Class>
9 <owl:ObjectProperty rdf:about="&site;teaches">
10  <rdfs:isDefinedBy rdf:resource="&site;"/>
11  <rdfs:domain rdf:resource="&site;academic"/>
12  <rdfs:range rdf:resource="&site;student"/>
13  <rdfs:label xml:lang="en"><![CDATA[teaches]]></rdfs:label>
14  <rdfs:comment xml:lang="en"><![CDATA[Academic teaches students]]></
      rdfs:comment>
15 </owl:ObjectProperty>
```

Programm 5.1: Ein Ausschnitt aus dem Quellcode der erzeugten Website-Ontologie, welche unter der Adresse http://domain.tld/rdf erreichbar ist. Deutlich erkennbar sind die Verknüpfungen als Hierarchie und Wertebereich.

es sich bei der editierten Eigenschaft um das Elternelement der eingegebenen Eigenschaft handelt. „Subproperty of“ deckt den inversen Fall ab, dass die editierte Eigenschaft von dem eingegebenen Elternelement abgeleitet ist. Beide Felder sind mit einer automatischen Vervollständigung ausgestattet, welche bereits bei den ersten Eingaben nach möglichen Begriffen sucht und diese zur Auswahl stellt. Mit den beiden Möglichkeiten die Hierarchie zu definieren, hat der Benutzer die Möglichkeit die Website-Ontologie entweder in der Top-Down- oder Bottom-Up-Methode (s. Abschnitt 4.1) zu erstellen.

Die weiteren Felder „Domain“ und „Range“ stehen lediglich für die Verarbeitung von Eigenschaften bereit und legen den Wertebereich eben dieser fest. Die automatische Vervollständigung stellt für beide Felder eine Auswahl von Klassen bereit, wobei für das Feld „Range“ zusätzlich XML-Datentypen mit dem Prefix `xsd:` unterstützt werden.

RDF/XML-Ausgabe

Die Ausgabe angelegter Klassen und Eigenschaften im RDF/XML-Format ist in Programm 5.1 dargestellt und erfolgt unter der vom Modul reservierten Adresse http://domain.tld/rdf. Diese Adresse kann nicht modifiziert werden.

Die Begriffe werden in ihre RDF bzw. OWL Äquivalente transformiert. Verknüpfungen in Form von Hierarchie- oder Wertebereich-Definitionen werden automatisch aufgelöst und korrekt ausgegeben.

5.4 Implementierung

In diesem Abschnitt werden als erstes der Aufbau des Moduls skizziert und eine kurze Erläuterung zu Drupal-Hooks (s. Abschnitt 5.4.2) gegeben. Anschließend sollen einige Hintergründe zur Umsetzung beleuchtet und anhand von Programmbeispielen vertieft werden.

5.4.1 Gliederung des Moduls

Das Modul wurde unter dem Kurznamen *siteonto* entwickelt und verwendet diesen als Prefix für Tabellen und Funktionsnamen sowie als Verzeichnisnamen. Die Website-Ontologie besteht, wie jedes Drupal-Modul, aus der obligatorischen Info-Datei und einer Reihe von optionalen Dateien. Zur Übersicht und besseren Abgrenzung wurden Funktionen in Include-Dateien ausgelagert. Die folgende Liste gibt einen Überblick über die einzelnen Dateien und deren Funktion.

siteonto.info Die obligatorische Datei für jedes Drupal-Modul beinhaltet den Namen, die Beschreibung sowie das eingeordnete Paket und die minimale Drupal-Version. Darüber hinaus werden Abhängigkeiten zu anderen Modulen, in diesem Fall RDFx und Evoc sowie die weiteren Dateien für das Modul aufgelistet.

siteonto.install Diese Datei wird während der Installation des Moduls aufgerufen und die darin befindlichen Anweisungen zum Anlegen der Tabellen ausgeführt werden. Anweisungen für mögliche Anpassungen der Tabellenstruktur werden ebenfalls an dieser Stelle abgelegt und während der Aktualisierung des Moduls abgearbeitet.

siteonto.module Definiert z. B. einige grundlegende Funktionen zum Überprüfen der Zugriffsrechte für bestimmte Aktionen oder Anlegen der Menüpunkte und URL zur Ausgabe der Website-Ontologie.

siteonto.admin.inc Bündelt Anweisungen für die Darstellung der Übersichtsliste von Klassen und Eigenschaften sowie die Formulare zum Editieren und Löschen selbiger.

siteonto.model.inc Kapselt Funktionen, welche auf die Datenbank zugreifen und z. B. Ressourcen und Verknüpfungen anlegen, editieren oder löschen.

siteonto.pages.inc Enthält die Anweisung zum Generieren der Website-Ontologie im RDF/XML-Format aus den Ressourcen und Verknüpfungen der Datenbank.

5.4.2 Drupal-Hooks

Die Programmierung von Modulen für Drupal wird durch die Verwendung der angebotenen Programmierschnittstellen (engl. Application Programming Interface , API) unterstützt. Drupal ist bis auf einige Basis-Funktionalitäten,

wie z. B. dem Bootloader, komplett modular aufgebaut. Bereits mit der Standardinstallation werden eine Reihe von Modulen z. B. für die Verwaltung von Inhaltstypen oder Benutzern ausgeliefert.

Die Philosophie von Drupal ist, dass jedes Modul eine differenzierte Funktionalität besonders gut implementiert und diese dann in Form von definierten Schnittstellen anderen Modulen zur Verfügung stellt. Durch die Vielzahl von Modulen entsteht eine Art Baukastensystem, in dem die Module aufeinander verweisen und folglich Abhängigkeiten eingehen. So ist es bei Drupal üblich, für ein neues Modul weitere abhängige Module zu suchen und vorweg zu installieren.

Drupal besitzt, gegenüber anderen CMS-Vertretern, keine klassenbasierte API, sondern verwendet die erwähnten Schnittstellen, die so genannten Hooks. Technisch gesehen handelt es sich bei Hooks um Funktionsaufrufe, die vom Modul-Autor an relevanten Stellen des Programmablaufs platziert werden. Während der Laufzeit verzweigt sich der Aufruf und die „eingehängte“ Funktion wird ausgeführt. Das Einhängen wird in Drupal über eine Namenskonvention der Funktionen gelöst, wodurch das CMS die Namen verwalten kann und bei dem Aufruf des Hooks entscheidet, welche Funktionen ausgeführt werden. Nachdem der Hook abgearbeitet wurde, kehrt das Programm in die ursprüngliche Funktion zurück, nimmt eventuelle Rückgabewerte entgegen und folgt dem weiteren Ablauf.

Dieses Verständnis über Hooks ist für die folgenden Programmbeispiele von Bedeutung, da auch die Implementierung in der vorliegenden Arbeit auf diversen Hooks basiert.

5.4.3 Datenbankschema

Die Klassen und Eigenschaften der Website-Ontologie sowie deren Verknüpfungen werden in der Datenbank der Drupal-Installation persistiert. Die verwendeten Tabellen sind im Datenbankschema in Abbildung 5.8 abgebildet und werden bei der Installation des Moduls angelegt. Die Tabellenstruktur wurde dazu bewusst einfach gehalten, um lediglich die notwendigen Funktionalitäten der Website-Ontologie abzubilden.

Die zentrale Tabelle `siteonto_terms` persistiert die Ressourcen der Ontologie. Dabei stellt jede Zeile innerhalb der Tabelle entweder eine Klasse oder Eigenschaft dar. Die nachfolgende Liste gibt eine kurze Erklärung zu den einzelnen Feldern der Tabelle.

type Die Art der Ressource (Klasse oder Eigenschaft).

enabled Ein boolescher Wert der angibt, ob die Ressource aktiviert ist und in der Website-Ontologie ausgegeben wird.

language Die Sprache für die Beschreibung des Elementes.

machine_name Der maschinenlesbare Name, welcher als Bezeichner der Ressource in der Website-Ontologie verwendet wird.

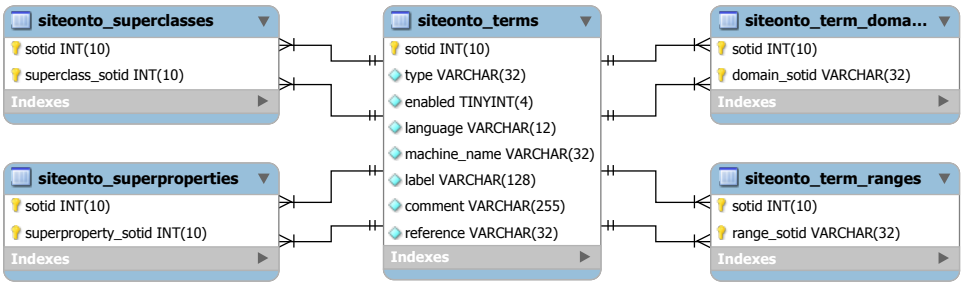

Abbildung 5.8: Datenbankschema des Website-Ontologie Moduls. Die Term-Tabelle persistiert Klassen und Eigenschaften. Verknüpfungen untereinander werden in den umliegenden Tabellen abgelegt.

label Der menschenlesbare Titel der Ressource.

comment Der Kommentar zur Ressource.

reference Referenz auf mögliche Inhaltstypen und deren Felder, sofern die Ressource daraus erzeugt wurde.

Bei den weiteren Tabellen handelt es sich um Verknüpfungstabellen einer selbst-referenzierenden m:n-Relation der `siteonto_terms` Tabelle. Diese speichern die Hierarchie der Klassen und Eigenschaften, sowie Domain- und Range-Verknüpfungen zwischen den Ressourcen. Dabei ist zu beachten, dass die Hierarchie-Tabellen `siteonto_superclasses` und `siteonto_superproperties` von der abgeleiteten Ressource zur Eltern-Ressource zeigen.

Eine weitere Besonderheit ist in der Tabelle `siteonto_term_ranges` verborgen. Das `range_sotid`-Feld wurde auf den Typ Varchar festgelegt, so dass neben den IDs aus der Term-Tabelle auch Datentypen in Form einer XSD-Zeichenkette abgelegt werden können.

Die Initialisierung der Tabellen in der Datenbank findet über die Schema-API von Drupal statt.

5.4.4 Transformation der Inhaltstypen

Direkt nach dem Anlegen der Tabellen wird der in Programm 5.2 abgebildete Install-Hook der Website-Ontologie ausgeführt. Der Aufruf `node_type_get_types()` gibt ein Array mit den verwendeten Inhaltstypen zurück. Diese werden vom inkludierten Model des Moduls als Klassen in der Datenbank eingetragen. Anschließend werden für jeden Inhaltstypen mit der Funktion `field_info_instances()` die zugehörigen Felder ausgelesen. Sofern diese existieren, werden sie ebenfalls durch das Model als Eigenschaften in der `siteonto_terms` Tabelle abgelegt und mit der Klasse über Domain und Range verknüpft.

Durch diese Funktion startet der Benutzer nicht mit einer leeren Website-

```
1 function siteonto_install() {
2   // include model to get insert functions
3   require_once DRUPAL_ROOT . '/' . drupal_get_path('module', 'siteonto')
        . "/siteonto.model.inc";
4
5   foreach (node_type_get_types() as $type => $name) {
6     $name->reference = 'content_type';
7     siteonto_node_type_insert($name);
8
9     $instances = field_info_instances('node', $name->type);
10    if(count($instances) > 0) {
11      foreach ($instances as $instance) {
12        siteonto_field_create_instance((object) $instance);
13      }
14    }
15  }
16 }
```

Programm 5.2: Der Install-Hook iteriert durch die existierenden Inhaltstypen mit deren Feldern und persistiert sie als Klassen und Eigenschaften in den angelegten Modul-Tabellen.

```
1 function siteonto_node_type_insert($info) {
2   $info->sotid = siteonto_get_tid_by_machine_name('class', $info->type);
3   // skip if something found
4   if($info->sotid !== FALSE) {
5     return FALSE;
6   }
7
8   $info->sotid = db_insert('siteonto_terms')
9     ->fields(array(
10      'type' => 'class',
11      'enabled' => 0, // disable by default!
12      'machine_name' => $info->type,
13      'language' => $GLOBALS['language']->language,
14      'label' => $info->name,
15      'comment' => $info->description,
16      'reference' => $info->module,
17    ))
18    ->execute();
19
20  return TRUE;
21 }
```

Programm 5.3: Die Funktion implementiert den Drupal-Hook `hook_node_type_insert()` und persistiert einen Inhaltstyp als Klasse der Website-Ontologie.

Ontologie, sondern findet bereits nach der Installation die Inhaltstypen als semantische Ressourcen wieder und kann von dieser Ausgangsbasis weitere Klassen und Eigenschaften definieren.

Nach der Transformation der existierenden Inhaltstypen bei der Installation des Moduls kann der Benutzer die Inhaltstypen nachträglich verändern. Diese Änderungen spiegeln sich nicht automatisch in der Website-Ontologie wider. Der Benutzer müsste folglich nach jeder Überarbeitung das Modul erneut installieren.

Als Lösung für diesen Umstand verwendet das Modul weitere Hooks, die beim Anlegen, Editieren oder Löschen von Inhaltstypen oder Feldern ausgeführt werden. In Programm 5.3 implementiert die Funktion `siteonto_node_type_insert()` den Drupal-Hook `hook_node_type_insert()`, welche als Parameter ein Objekt mit Informationen zum erzeugten Inhaltstyp erhält. Wurde dieser Inhaltstyp bereits als Klasse in der Website-Ontologie abgelegt, wird die Funktion verlassen. Andernfalls wird der Inhaltstyp als neue Klasse in der Term-Tabelle abgelegt. Dabei ist zu beachten, dass die neue Ressource standardmäßig deaktiviert und somit vor einer versehentlichen Veröffentlichung geschützt ist. Für das Bearbeiten und Löschen von Inhaltstypen stehen mit `hook_node_type_update()` und `hook_node_type_delete()` weitere Hooks zur Verfügung, die vom Modul implementiert werden.

Die Hooks `hook_field_create_instance()`, `hook_field_update_instance()` und `hook_field_delete_instance()` für die Felder funktionieren äquivalent zu den Inhaltstypen. Einziger Unterschied ist, dass in der implementierten Funktion automatisch eine Domain-Range-Verknüpfung zur passenden Klasse in der Datenbank abgelegt und beim Löschen des Feldes zusammen mit diesem auch wieder entfernt wird.

Durch die implementierten Hooks ist die Website-Ontologie mit jeder Änderung der Inhaltstypen ohne Verzögerung auf dem aktuellen Stand. Der Benutzer hat somit die Möglichkeit, die Inhaltstypen als Klassen und Eigenschaften einfach in die Entwicklung der Ontologie einzubeziehen.

5.4.5 Formulargenerierung

In der Übersicht werden dem Benutzer alle erstellten Klassen und Eigenschaften der Ontologie aufgelistet. Sollen nun weitere Ressourcen hinzugefügt oder bestehende editiert werden, so wechselt die Ansicht zum Formular, welches in Abb. 5.6 dargestellt ist.

Für die Implementierung des Formulars wurde die Drupal Form API verwendet. Als Schnittstelle fungiert die Funktion `drupal_get_form()`, welche die Formularausgabe aus einem Array mit Anweisungen für die Formularfelder generiert. Die Drupal-Funktion verlangt dazu den Namen einer weiteren Funktion, die das Array erstellt und zurückgibt. Die Verwendung der Form API umfasst drei Schritte, die in einzelnen Funktionen gekapselt und ausgeführt werden.

```
1 $form['label'] = array(
2   '#title' => t('Label'),
3   '#type' => 'textfield',
4   '#default_value' => $edit->label,
5   '#required' => TRUE,
6   '#size' => 30,
7 );
8
9 $form['machine_name'] = array(
10   '#title' => t('Machine name'),
11   '#type' => 'machine_name',
12   '#default_value' => $edit->machine_name,
13   '#maxlength' => 32,
14   '#machine_name' => array(
15     'source' => array('label'),
16     'replace_pattern' => '[^a-z0-9_]+',
17     'replace' => '_',
18     'exists' => (($type == 'class') ? 'siteonto_class_exists' : '
       siteonto_property_exists'),
19   ),
20   '#disabled' => FALSE,
21 );
```

Programm 5.4: Definition der Formularfelder Label und Maschinenname. Der Maschinenname konvertiert das Label automatisch in Kleinbuchstaben und filtert Sonderzeichen heraus.

Formulargenerierung Erstellt ein assoziatives Array mit Formularfeldern, wobei jedes Feld wiederum spezielle Anweisungen für die Formatierung und das Verhalten enthält.

Validierung Sendet der Benutzer seine Formulareingaben ab, wird die Validierung angestoßen. Stimmen die Eingaben nicht mit den Vorgaben überein, kann eine Fehlermeldung ausgegeben werden und das Formular wird erneut angezeigt.

Verarbeitung Wurden die Eingaben erfolgreich validiert, können diese verarbeitet werden. In den häufigsten Fällen besteht die Verarbeitung aus der Persistierung in der Datenbank.

Die Formulargenerierung wurde in der Funktion `siteonto_form_edit()`, die Validierung in `siteonto_form_edit_validate()` und die Verarbeitung in `siteonto_form_edit_submit()` implementiert. Dabei ist zu beachten, dass die Validierungsfunktion für den Prototypen nicht umgesetzt ist.

Im Folgenden sollen einige Stellen aus der Formulargenerierung und der Datenverarbeitung näher betrachtet werden.

Das Programm 5.4 stellt die Definition für die Formularfelder Label und Maschinennamen im assoziativen Array dar. Beide Felder verfügen über einen Titel, Typ und Standardwert, welcher beim Bearbeiten aus der Daten-

```
1 $form['fieldset_relations']['subclassof'] = array(
2   '#type' => 'textfield',
3   '#title' => t('Subclass of'),
4   '#default_value' => (isset($edit->sotid)) ?
      siteonto_get_relations_implode($type, 'siteonto_superclasses', '
      superclass_sotid', array('field'=>'sotid', 'value'=>$edit->sotid)) :
       '',
5   '#autocomplete_path' => 'admin/structure/siteonto/class/autocomplete',
6 );
```

Programm 5.5: Definition eines Formularfeldes mit automatischer Vervollständigung. Die Daten werden während der Benutzereingabe beim definierten Pfad angefragt.

bank ausgelesen wird. Die Besonderheit befindet sich im Maschinenname, welcher den Wert eines verknüpften Feldes nach einem definierten Muster bereinigt. Im vorliegenden Beispiel wird das Label in Zeile 15 als Quelle definiert und automatisch in Kleinbuchstaben konvertiert. Sonderzeichen werden gegen einen Unterstrich ersetzt (Zeile 16 und 17). Zusätzlich werden die Eingaben des Benutzers in der Formularansicht, ohne die komplette Seite nachzuladen, auf bereits existierende Elemente überprüft, wobei der Funktionsname nach Klassen und Eigenschaften in Zeile 18 unterschieden wird. Drupal erstellt aus dieser Anweisung automatisch die passende Ausgabe für die Formularfelder.

Eine weitere Besonderheit sind die Formularfelder für Super- und Subelemente, sowie Domain- und Range-Einschränkungen, welche eine automatische Vervollständigung der angefangenen Benutzereingabe bietet (s. Abb. 5.7). Alle Felder folgen der gleichen Implementierung und unterscheiden sich lediglich vom Typ der angefragten Daten. Das Programm 5.5 stellt die Definition für das Formularfeld der Subklasse im assoziativen Array dar. In den Zeilen 3 und 4 werden Typ und Titel festgelegt. Die darauf folgende Zeile legt den Standardwert fest. Editiert der Benutzer einen Begriff, werden alle bestehenden Verknüpfungen für den Begriff als kommaseparierte Liste von der Funktion `siteonto_get_relations_implode()` zurückgegeben. Die Zeile 5 gibt den Pfad für die automatische Vervollständigung an.

Die aufgerufene Seite bzw. Funktion ist in Programm 5.6 dargestellt und bekommt als Parameter den Typ des Elementes, sowie die bisherige Eingabe des Textfeldes als Zeichenkette. Die Eingabe wird in den nachfolgenden Zeilen am Komma getrennt und die letzte Eingabe in der Variable $terms_last abgelegt. Beginnt diese Zeichenkette mit „xs“ so wird ein XML-Datentyp erwartet und in Zeile 8 eine Reihe an Datentypen als Ergebnis zurückgegeben. Andernfalls wird in der Datenbank nach Begriffen mit den eingegebenen Anfangsbuchstaben gesucht und die Ergebnisse werden in Zeile 17 für die Ausgabe hinzugefügt. Dabei ist zu beachten, dass der Schlüssel des Arrays den

```
1 function siteonto_term_autocomplete($type, $string) {
2   $terms_typed = drupal_explode_tags($string);
3   $terms_last = drupal_strtolower(array_pop($terms_typed));
4   $matches = array();
5   if ($terms_last != '') {
6     $terms_entered = count($terms_typed) ? implode(', ', $terms_typed) .
       ', ' : '';
7     if(stripos($terms_last, 'xs') === 0) {
8       $matches = siteonto_datatype_autocomplete($terms_last);
9     } else {
10      $result = db_select('siteonto_terms', 'sot')
11          ->fields('sot', array('sotid', 'machine_name', 'label'))
12          ->condition('machine_name', $terms_last.'%','LIKE')
13          ->condition('type', $type, '=')
14          ->execute();
15      foreach ($result as $row) {
16        $row->curie = 'site';
17        $matches[$terms_entered.$row->machine_name] = theme('
       siteonto_term_autocomplete', array('term' => $row,));
18      }
19    }
20  }
21  drupal_json_output($matches);
22 }
```

Programm 5.6: Die Funktion trennt die kommaseparierte Zeichenkette auf und sucht nach den letzten Eingaben in der Datenbank. Die Ergebnisse werden als JSON-Objekt zurückgegeben.

Text für das Formularfeld enthält und der Wert über die Drupal-Funktion `theme()` formatiert wird. In Zeile 21 werden die Ergebnisse durch die Funktion `drupal_json_output()` in ein JSON-Objekt umgewandelt und direkt ausgegeben.

5.4.6 RDF/XML-Ausgabe

Die angelegten Begriffe werden über die URL „domain.tld/rdf“, welche in der Funktion `siteonto_menu()` im CMS vom Modul registriert wurde, ausgegeben. Die Ausgabe im RDF/XML-Format ist in Programm 5.1 dargestellt. Die Konvertierung der Begriffe ist in der Funktion `siteonto_generate_rdf()` implementiert.

Das Programm 5.7 erstellt den Kopf der Website-Ontologie. In der ersten Zeile wird der MIME-Typ des Dokumentes auf RDF/XML umgestellt, wodurch die Identifizierung und Verarbeitung für externe Programme erleichtert wird. Die nachfolgenden Programmzeilen definieren beispielhaft zwei Namensräume. Zum einen den Pfad zur Ausgabe der Website-Ontologie und zum anderen den Pfad für RDF. Weitere Namensräume müssen derzeit hän-

```
1 header('Content-Type: application/rdf+xml; charset=UTF-8');
2
3 $namespaces['site'] = array('curie' => 'site', 'uri' => url('', array('
      absolute' => TRUE)).'rdf#');
4 $namespaces['rdf'] = array('curie' => 'rdf', 'uri' => 'http://www.w3.org
      /1999/02/22-rdf-syntax-ns#');
5 // define further namespaces
6
7 $output = $ns_string = '';
8 $output .= '<?xml version="1.0" encoding="UTF-8"?>\n';
9 $output .= '<!DOCTYPE rdf:RDF [\n';
10 foreach($namespaces as $curie => $ns) {
11    $output .= '\t<!ENTITY '.$ns['curie'].' "'.$ns['uri'].'">\n';
12    $ns_string .= ' xmlns:'.$ns['curie'].'="&'.$ns['curie'].';"';
13 }
14 $output .= ']>\n';
15 $output .= '<rdf:RDF'.$ns_string.'>\n';
16 $output .= '<owl:Ontology rdf:about="&site;"><rdfs:label>Website
      Ontology</rdfs:label></owl:Ontology>\n';
```

Programm 5.7: Erstellt den Kopf der Website-Ontologie als RDF/XML-Ausgabe.

disch hinzugefügt werden. In den Zeilen 9-14 werden die definierten Namensräume als XML-Entitäten definiert, so dass die Namensräume weiter unten im Dokument nicht mit der kompletten URI, sondern mit der kompakten URI (CURIE[3]) angesprochen werden können. Zeitgleich werden in der Schleife die XML-Namensraum-Attribute zusammengebaut und dem RDF-Tag in Zeile 15 zugewiesen. Es folgt eine Beschreibung der vorliegenden Ontologie mit dem menschenlesbaren Label. Die gesamte Ausgabe wird während der Generierung in der Variable `$output` gespeichert und am Ende der Funktion ausgegeben.

Die Eigenschaften der Website-Ontologien werden im Programm 5.8 in das RDF/XML-Format konvertiert. Für die Klassen ist der Ablauf nahezu identisch und wird daher nicht gesondert angeführt. Mit der ersten Zeile werden alle aktivierten Eigenschaften ausgelesen und anschließend nacheinander durchlaufen. Zu jeder Eigenschaft werden in Zeile 3 die Range-Verknüpfungen aus der Datenbank ausgelesen. In den nachfolgenden Zeilen 4-9 werden die ermittelten Wertebereiche durchlaufen und auf einen XML-Datentyp untersucht. Wird kein Treffer gefunden, handelt es sich bei der aktuellen Eigenschaft um eine `owl:ObjectProperty`, andernfalls um eine `owl:DatatypeProperty` (s. Abschnitt 2.2.3). Anschließend wird die Eigenschaft definiert und der Website-Ontologie mit `rdfs:isDefinedBy` zugeschrieben. Sind Range-Verknüpfungen vorhanden, so werden diese in den

[3] http://www.w3.org/TR/curie/

```
1 $result = siteonto_load_multiple('property', array(), array(), array('
      field' => 'enabled', 'value' => 1, 'operator' => '='));
2 foreach ($result as $row) {
3   $ranges = siteonto_get_relations('class', 'siteonto_term_ranges', '
      range_sotid', array('field' => 'sotid', 'value' => $row->sotid));
4   $tag = 'owl:ObjectProperty';
5   foreach($ranges as $range) {
6     if(siteonto_datatype_exists($range) == TRUE) {
7       $tag = 'owl:DatatypeProperty'; break;
8     }
9   }
10  $output .= '<'.$tag.' rdf:about="&site;'.$row->machine_name.'">\n';
11  $output .= '\t<rdfs:isDefinedBy rdf:resource="&site;"/>\n';
12  if(!empty($ranges)) {
13    foreach($ranges as $machine_name) {
14      if(stripos($machine_name, 'xs') === 0) {
15        $output .= '\t<rdfs:range rdf:resource="'.str_replace('xsd:', '&
      xsd;', $machine_name).'"/>\n';
16      } else {
17        $output .= '\t<rdfs:range rdf:resource="&site;'.$machine_name.'
      "/>\n';
18      }
19    }
20  }
21  if(strlen($row->label) > 0) {
22    $output .= '\t<rdfs:label xml:lang="'.$row->language.'"><![CDATA['.
      $row->label.']]></rdfs:label>\n';
23  }
24  $output .= '</'.$tag.'>\n';
25 }
```

Programm 5.8: Generiert die Eigenschaften der Website-Ontologie und unterscheidet dabei zwischen `owl:ObjectProperty` und `owl:DatatypeProperty`. Des Weiteren wird beispielhaft der Wertebereich und das Label ausgegeben.

Zeilen 12-20 nacheinander durchlaufen und der Maschinenname auf einen XML-Datentyp untersucht. Im Falle eines Fundes wird der passende XSD- bzw. Site-Namensraum bei keinem Treffer als Entität vorangestellt. Alle weiteren Verknüpfungen wie `rdfs:domain` oder `rdfs:subPropertyOf` werden in äquivalenter Weise erstellt. In den Zeilen 21-23 wird, sofern vorhanden, das menschenlesbare Label mit der passenden Sprache ausgegeben. Dabei ist zu beachten, dass der Text mit einem CDATA umschlossen wurde, um Probleme mit Sonderzeichen zu vermeiden. Diese Implementierung wird für die Ausgabe des Kommentars in gleicher Weise wiederholt.

```
1 function siteonto_import_rdfx_data() {
2   if(module_exists('rdfx') == FALSE || module_exists('evoc') == FALSE) {
3     return FALSE; // abort if modules not exists
4   }
5   $site_uri = url('', array('absolute' => TRUE)).'rdf#';
6   evoc_import_vocabulary($site_uri, 'site');
7 }
```

Programm 5.9: Importiert die Website-Ontologie mit Hilfe des Evoc-Moduls erneut. Dazu wird lediglich die komplette URI und die kompakte URI an die Funktion übergeben.

5.4.7 RDFx-Anbindung

Eine Anbindung der Website-Ontologie an die bestehenden RDFx-Module schließt den Kreis von der Erstellung hin zur Verwendung der Ontologie.

Die vorher beschriebene Implementierung zur Website-Ontologie ist komplett unabhängig von den RDFx-Modulen. Die Implementierung hat zum Ziel die RDFx-Module frei von Modifikationen zu halten, da mögliche Änderungen von den Autoren aufgenommen werden müssen. Als Lösung wurde daher eine Brücke ausgehend von der Website-Ontologie zu den fremden Modulen geschlagen, die im wesentlichen auf zwei Pfeilern beruht.

Das Programm 5.9 stellt den ersten Teil dar, welcher überprüft, ob die RDFx-Module und das Evoc-Modul existieren. Anschließend wird das Evoc-Modul genutzt, um die Website-Ontologie von der öffentlich erreichbaren URI mit dem „site“-Prefix zu importieren. Nach einem erfolgreichen Import befindet sich anschließend eine Kopie der Website-Ontologie in den RDFx-Tabellen. Auf dieser Basis kann der Benutzer die Inhaltstypen mit allen öffentlichen Klassen und Eigenschaften der Website-Ontologie annotieren (s. Abschnitt 5.2.1).

Der zweite Teil für die Verbindung zum RDFx-Modul ist in Programm 5.10 dargestellt. In Zeile 2-4 wird anhand der öffentlichen URI der Website-Ontologie die passende Graph-ID aus den RDFx-Tabellen ermittelt, die in der nachfolgenden Zeile 6 dazu verwendet wird alle verwendeten IDs der Namensräume auszulesen. Anschließend werden für jeden Namensraum in den Zeilen 8-10 die abhängigen Begriffe (Klassen und Eigenschaften) ausgelesen und zwischengespeichert. Mit den gesammelten Informationen wird anschließend jede RDFx-Tabelle durchlaufen und die Begriffe mit deren Verknüpfungen werden entfernt. Abschließend werden ab Zeile 20 die Namensräume und der Graph gelöscht. Damit sind alle Informationen der Website-Ontologie, die von den RDFx-Modulen gehalten werden, entfernt.

Beide Programme werden bei Modifikation der Website-Ontologie nacheinander ausgeführt. Als erstes werden mögliche bestehende Daten aus den RDFx-Modulen gelöscht und anschließend die aktualisierten Klassen und

```
1 $nsids = $tids = array();
2 $site_uri = url('', array('absolute' => TRUE)).'rdf#';
3 $gid = rdfx_get_gid($site_uri);
4 if($gid == 0) return FALSE; // abort condition
5
6 $nsids = db_query("SELECT uri, nsid FROM {rdfx_namespaces} WHERE gid = :
      gid", array(':gid' => $gid))->fetchAllKeyed();
7 if(count($nsids) == 0) return FALSE; // abort condition
8 foreach($nsids as $ns => $nsid) {
9   $tids[$nsid] = array_values(db_query("SELECT tid FROM {rdfx_terms}
      WHERE nsid = :nsid", array(':nsid' => $nsid))->fetchCol());
10 }
11
12 foreach($tids as $nsid => $tid) {
13   if(count($tid) == 0) continue;
14   db_delete('rdfx_terms')
15    ->condition('tid', $tid, 'IN')
16    ->execute();
17   // delete tid in further rdfx tables
18 }
19
20 db_delete('rdfx_namespaces')
21  ->condition('gid', $gid)
22  ->execute();
23 // delete gid in further rdfx tables
```

Programm 5.10: Ermittelt anhand der Website-Ontologie-URI die ID des Graphen, sowie die IDs der verwendeten Namensräume im RDFx Modul. Anschließend werden alle Begriffe, Namensräume und Graphen aus den RDFx-Tabellen entfernt.

Eigenschaften hinzugefügt.

5.5 Szenario

Ausgehend von der Implementierung aus dem vorangegangenen Kapitel soll diese in Form eines Szenarios verwendet werden, um eine Website-Ontologie zu erstellen. In einem ersten Schritt werden dazu die Rahmenbedingungen abgesteckt und nachfolgend eine Beispielseite erstellt.

5.5.1 Voraussetzungen

Damit die Implementierung getestet werden kann, wird in diesem Schritt beispielhaft eine Internetseite der fiktiven New Town Universität eingerichtet. Die grundlegende Struktur der Seite ist in Form einer Sitemap in Abbildung 5.9 dargestellt. Die Seite unterteilt sich in die vier Hauptbereiche: Studiengänge, Vorlesungen, Mitarbeiter und Stellenangebote. Jeder Bereich besitzt

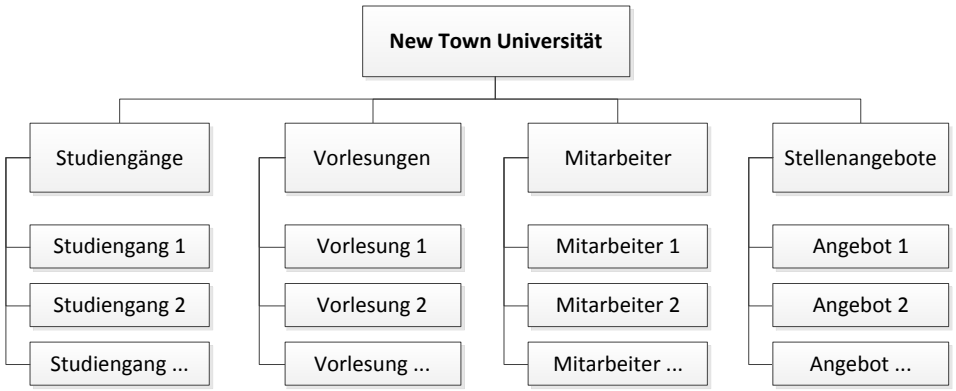

Abbildung 5.9: Die Sitemap der fiktiven New Town Universität.

wiederum einzelne Unterseiten.

Für die Universitätsseite wird eine vollständige Standardinstallation von Drupal 7 verwendet. Damit z. B. einige Übersichtsseiten umgesetzt werden können, werden zusätzliche Module manuell installiert. Die folgende alphabetisch sortierte Liste stellt die wichtigsten verwendeten Module dar.

ctools Die Chaos Tool Suite stellt eine Sammlung von nützlichen Funktionen für die Entwicklung weiterer Module zur Verfügung.

entity Erweitert die Entity API von Drupal und stellt einen vereinfachten CRUD-Zugriff[4] für die einzelnen Entitäten von Drupal zur Verfügung.

devel Erleichtert die Entwicklung mit Drupal durch gezieltes Debugging von Nodes oder Reinstallieren von Modulen.

field_group Gruppiert die einzelnen Felder der Inhaltstypen und präsentiert diese somit in einer übersichtlichen Form für den Benutzer.

panels Hilft bei der Umsetzung flexibler Seiten, indem verschiedene Bereiche definiert und mit unterschiedlichen Inhalten gefüllt werden können.

references Bietet mit der Node- und Benutzer-Referenz zwei neue Felder für Inhaltstypen an.

styles Bündelt die Style-Angaben für verschiedene Felder und bietet eine API für andere Module an.

views Baut Datenbankabfragen mit Hilfe einer graphischen Benutzeroberfläche zusammen. Die formatierten Ergebnisse können anschließend mit Panels positioniert werden.

Selbstverständlich werden auch die RDFx-Erweiterung und die Website-Ontologie installiert. Hinweise zur Installation können in Abschnitt 5.3 nachgeschlagen werden.

[4]Die Abkürzung CRUD steht für Create, Read, Update and Delete. Damit deckt CRUD die häufigsten Aktionen, die auf einem Datensatz ausgeführt werden können, ab.

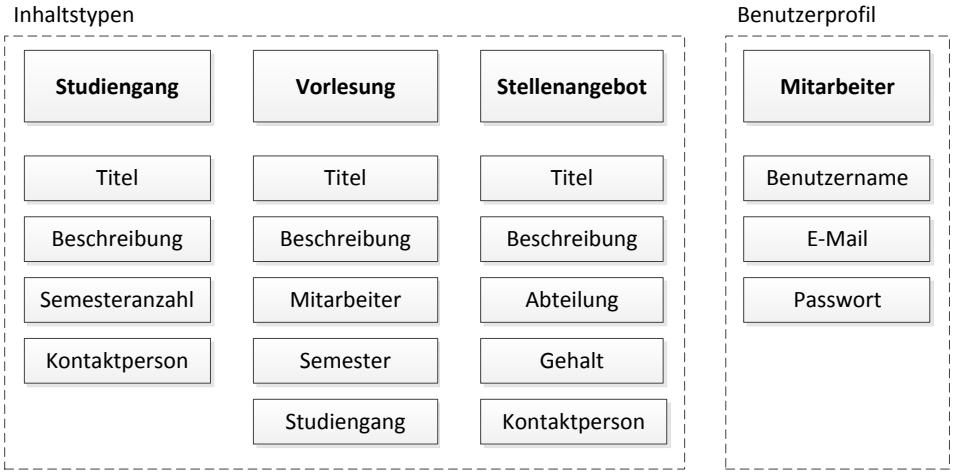

Abbildung 5.10: Verwendete Inhaltstypen mit den zugewiesenen Feldern sowie das Drupal-Benutzerprofil für die Mitarbeiter der Universität.

5.5.2 Umsetzung der Website

In einem ersten Schritt werden die benötigten Inhaltstypen konzipiert. Im Fall der Universitätsseite wird für jeden Bereich ein eigener Inhaltstyp verwendet. Die Abbildung 5.10 stellt die Inhaltstypen mit den zugehörigen Feldern dar. Eine Ausnahme stellen die Mitarbeiterseiten dar, welche nicht über die Standard-Benutzerprofile von Drupal erzeugt werden. Durch diese Verwendung wird die Handhabung und Verknüpfung mit den anderen Inhaltstypen z. B. als Kontaktpersonen erleichtert.

Die Inhaltstypen werden im gleichnamigen Punkt im Administrationsmenü unter Struktur angelegt. Zu jedem Inhaltstyp werden die konzipierten Felder hinzugefügt und abgespeichert. Auf die Zuweisung von RDF-Daten wird erst in einem späteren Arbeitsschritt eingegangen.

Im nächsten Schritt wird die Seite mit Inhalt gefüllt. Damit die Inhalte nicht händisch eingepflegt werden müssen, wird auf das Devel-Modul zurückgegriffen, welches für jeden Inhaltstyp die gewünschte Anzahl an Instanzen erstellt und diese mit zufälligen Werten füllt. Diese sind nicht besonders aussagekräftig, reichen jedoch um einen lebendigen Eindruck von der Beispielseite zu vermitteln.

Die Abbildung 5.11 stellt die passenden Einstellungen für die Inhaltstypen dar, welche im Administrationsmenü unter Konfiguration » Entwicklung » „Inhalte generieren“ vorgenommen und anschließend in die Datenbank eingetragen werden. Anschließend wird der Vorgang für die Benutzerprofile wiederholt. Äquivalent können über diesen Weg auch Begriffe, Menüs oder Vokabulare generiert werden.

Mit den synthetisch erzeugten Inhalten können im dritten Schritt die

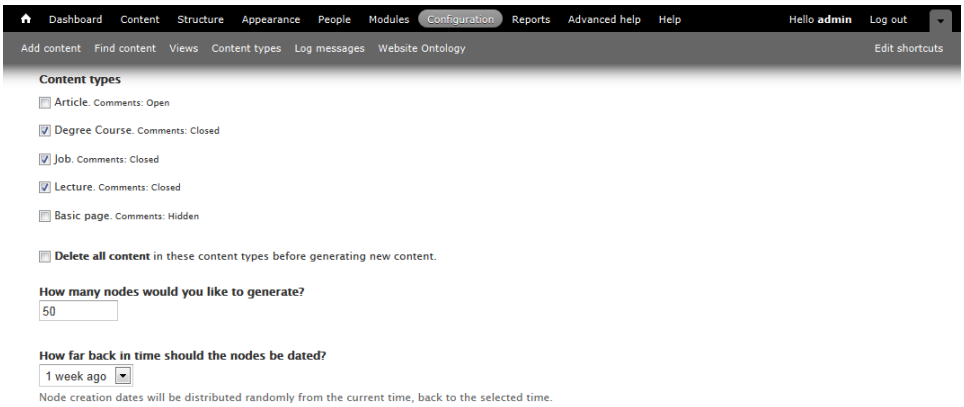

Abbildung 5.11: Beispielinhalte werden zufällig mit dem Devel-Modul generiert und persistiert. Äquivalent dazu werden die Benutzerprofile erzeugt.

Übersichtsseiten aus der Sitemap generiert werden. Mit Hilfe der graphischen Benutzeroberfläche des Views-Moduls wird eine Datenbankabfrage erstellt und die Ergebnisse in verschiedenen Ansichten präsentiert.

Die Abbildung 5.12 stellt im oberen Teil die Einstellungen für die Übersicht der Vorlesungen dar. Dort werden die auszulesenden Felder ausgewählt sowie Kriterien zum Filtern und Sortieren der Ergebnisse definiert. In der mittleren Spalte ist der Pfad für den Zugriff und ein Punkt im Menü angelegt. Im unteren Bereich der Abbildung ist ein Teil der Vorlesungen als Tabelle formatiert. Die Einstellung ist so gewählt, dass sich die Spalten über den Tabellenkopf sortieren lassen. Wählt der Benutzer den Studiengang, die Vorlesung oder den Mitarbeiter aus, so gelangt er jeweils zur Detailansicht, in welcher weitere Informationen hinterlegt sind.

Die weiteren Übersichtsseiten sind ähnlich der Vorlesungsseite aufgebaut und stellen die Daten ebenfalls in einer Tabelle dar. Lediglich für die Stellenangebote ist noch eine Auswahl für Abteilungen vorgeschaltet, so dass die Ergebnismenge in der Tabelle kleiner ausfällt.

Die Seite wurde einfach gehalten, da eine komplett ausgearbeitete Seite nicht das Ziel des Beispiels ist. So wurde z. B. das Aussehen nicht angepasst und das Theme „Bartik“ verwendet, welches mit Drupal 7 ausgeliefert wird.

5.5.3 Aufbau der Website-Ontologie

Nachdem die Internetseite der fiktiven Universität funktional fertiggestellt ist, soll in diesem Abschnitt der semantische Aspekt betrachtet werden. Dazu wird in einem ersten Schritt die Website-Ontologie aufgebaut.

Die Idee der Website-Ontologie ist es, das Wissen zum Themenbereich der Website in einer maschinenlesbaren Sprache zu formulieren. Die Ontologie für die Beispielseite soll die Struktur der Universität in stark vereinfachter

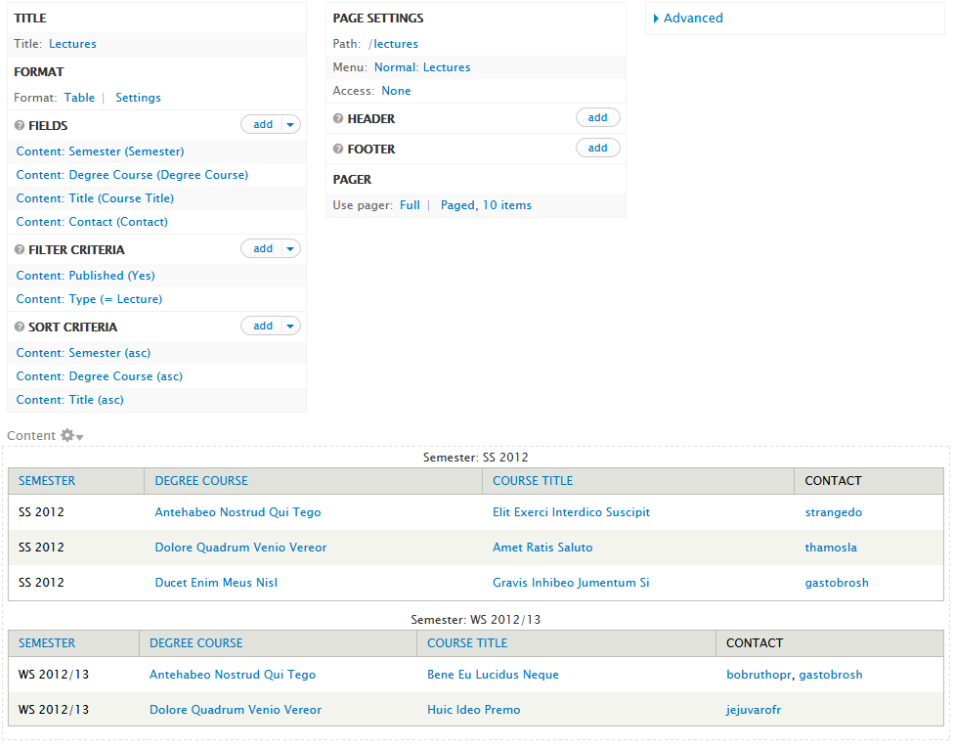

Abbildung 5.12: Die Einstellungen für eine Übersichtsseite im Views-Modul (oben) und die dazugehörige Vorschau der formatierten Ergebnisse (unten).

Form abbilden. Die Abbildung 5.13 veranschaulicht die geplante Umsetzung. Deutlich erkennbar sind die konvertierten Inhaltstypen (hellgrau) und die benutzerdefinierten Klassen (dunkelgrau). Die Eigenschaften sind als gerichtete Graphen dargestellt und mit dem Bezeichner beschriftet.

Auf der linken Seite der Abbildung wird definiert, dass die Universität ein oder mehrere Fachbereiche enthält. Diese bestehen wiederum aus einem oder mehreren Studiengängen usw. Die Differenzierung findet jeweils durch die benutzerdefinierte Eigenschaft `site:isPartOf` statt.

Die rechte Seite baut eine Hierarchie für Personen durch die Eigenschaft `rdfs:subClassOf` auf. Alle Mitarbeiter und Studenten unterstehen dabei der Klasse Person. Zusätzlich werden Mitarbeiter in Dozenten und Professoren unterschieden.

Beide Seiten werden mit weiteren benutzerdefinierten Eigenschaften, wie z. B. `site:publishes` oder `site:lectures` untereinander verknüpft. Dabei ist es ohne Bedeutung, ob es sich um einen konvertierten Inhaltstyp oder eine benutzerdefinierte Klasse handelt.

Alle abgebildeten Eigenschaften der Website-Ontologie (mit dem site-

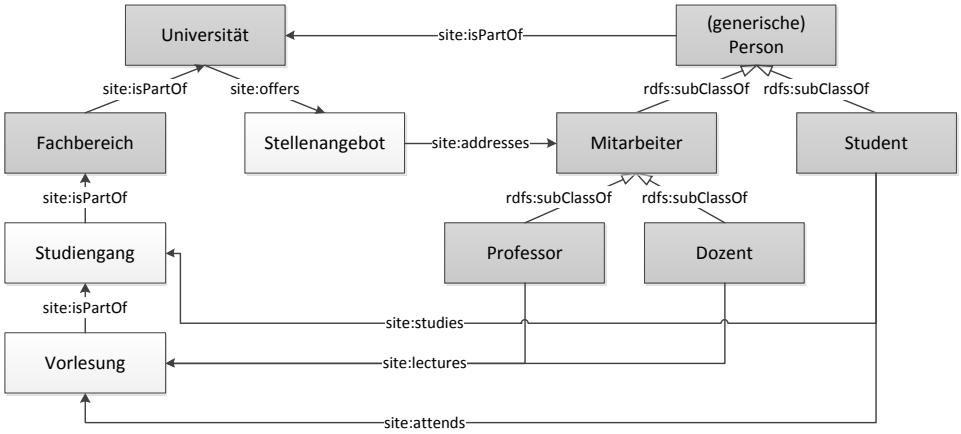

Abbildung 5.13: Schematische Darstellung der Website-Ontologie mit konvertierten Inhaltstypen (hellgrau) und benutzerdefinierten Klassen (dunkelgrau).

Prefix) werden in die `owl:ObjectProperty` umgewandelt und ausgegeben. Die `owl:DatatypeProperty`, welche ein Literal in Form eines XML-Datentyp erwarten, sind aufgrund der Übersichtlichkeit nicht abgebildet.

Umsetzung der Website-Ontologie

Für die Umsetzung sollten unter Struktur » Website-Ontologie im Administrationsmenü bereits die konvertierten Inhaltstypen als Klassen und die Felder als Eigenschaften gelistet sein. Allgemeine Informationen und Abbildungen zur Funktionsweise des Moduls können in Abschitt 5.3 nachgeschlagen werden.

Bei den benutzerdefinierten Ressourcen empfiehlt es sich mit den Klassen zu beginnen, da diese anschließend für die Erstellung der Eigenschaften vorausgesetzt werden. Über den Link „Add Class“ gelangt der Benutzer zum Eingabeformular, mit dem die weiteren Klassen hinzugefügt werden können. Als minimale Anforderung muss der Benutzer das Label ausfüllen. Der Maschinenname wird daraus automatisch generiert, kann optional jedoch geändert werden. Als Verknüpfungen können an dieser Stelle die abgebildeten `rdfs:subClassOf`-Beziehungen als Eltern- bzw. Kindklassen deklariert werden.

Im nächsten Schritt werden die Eigenschaften über das Eingabeformular hinzugefügt. Es werden die gleichen Angaben wie bei den Klassen verlangt. Zusätzlich wird der Wertebereich über die Domain- und Range-Felder durch eine Klasse oder ein XML-Datentyp eingeschränkt. Eine Ausnahme stellt die Eigenschaft `rdfs:isPartOf` dar, welche unterschiedliche Klassen als Wertebereich einschließt. Daher kann als Wertebereich lediglich eine übergeordnete

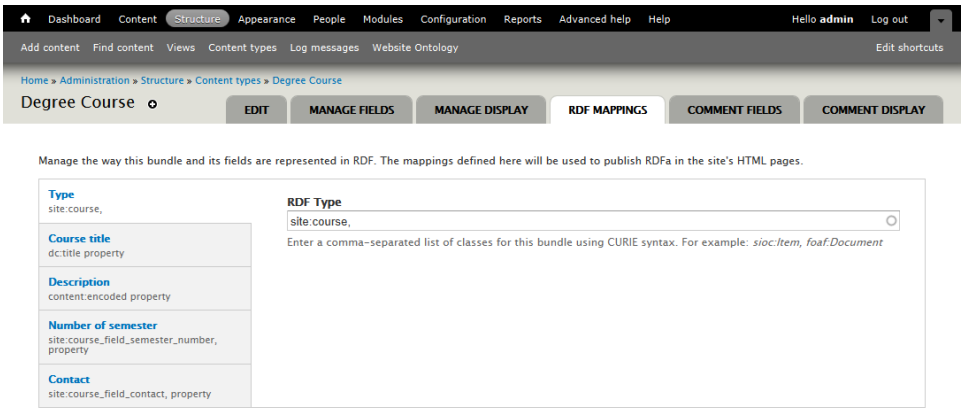

Abbildung 5.14: Mit Hilfe des RDFx-Moduls werden den Inhaltstypen die Klassen und Eigenschaften der Website-Ontologie zugewiesen.

Klasse, wie z. B. `owl:Thing`, oder keine Klasse angegeben werden.

Im letzten Schritt können nicht verwendete Klassen und Eigenschaften in der Übersicht ausgewählt und deaktiviert (oder bei Bedarf auch wieder aktiviert) werden.

Verwendung der Website-Ontologie

Nachdem die Website-Ontologie mit ihren Klassen und Eigenschaften angelegt wurde, wird diese aufbereitet und unter dem Pfad domain.tld/rdf/ für die weitere Verarbeitung zur Verfügung gestellt. Über das RDFx-Modul lassen sich die Ressourcen innerhalb von Drupal ohne Umwege wiederverwenden. Die Abbildung 5.14 stellt den Reiter „RDF Mapping“ für einen Inhaltstyp dar. Der Reiter kann im Administrationsmenü unter Struktur » Inhaltstypen aufgerufen werden. Anschließend können für jeden Inhaltstyp die Klassen und Eigenschaften über die automatische Vervollständigung ausgewählt und zugewiesen werden. Jede Zuweisung wird von Drupal automatisch als RDFa direkt in den HTML-Quelltext der Website gerendert und verweist auf die Website-Ontologie.

Kapitel 6

Evaluierung

Dieses Kapitel evaluiert die vorhergehende Implementierung der Website-Ontologie[1] für Drupal in Form eines Vergleiches mit den RDFx-Modulen (Version 6 und 7) und *Protégé*[2], einem eigenständigen Ontologie-Editor ohne CMS-Anbindung. Protégé wird unter der Leitung der Stanford University als Java-Anwendung entwickelt und als Open-Source-Anwendung veröffentlicht.[3]

Alle Benutzeraktionen und Funktionen der zu vergleichenden Programmen wurden gruppiert und stellen im Folgenden die Abschnitte dieses Kapitels dar.

6.1 Wiederverwendung bestehender Ontologien

Wie bereits im zweiten Punkt der Abbildung 4.2 dargestellt, sollte vor der Erstellung einer neuen Ontologie überprüft werden, ob eine existierende Ontologie wiederverwendet werden kann. Deckt sich eine Ontologie mit dem Anwendungsfall, so muss diese in die eigene Ontologie einbezogen werden, um z. B. weitere Klassen und Eigenschaften davon abzuleiten.

RDFx-Modul

Die Tabelle 6.1 vergleicht die Aktionen zum Importieren von gesamten Vokabularen und Ontologien, sowie die Wiederverwendung einzelner Ressourcen daraus. In Drupal können mit dem Evoc-Modul aus dem RDFx-Paket fremde Ontologien importiert und anschließend die Inhaltstypen mit den einzelnen Ressourcen annotiert werden. Der Import verarbeitet Hierarchien von

[1] Die Website-Ontologie wird in den nachfolgenden Tabellen unter dem Kurznamen *siteonto* geführt.

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

[3] http://protege.stanford.edu/download/download.html, Kopie auf CD-ROM (Datei onlinequellen/2011_stanford_protege-license.pdf vom 20.09.2011).

	RDFx v6 / v7	*Protégé*	*Siteonto*
Import von Vokabularen und Ontologie	Ja / Ja	Ja	Nein
Verwendung importierter Ressourcen	Ja / Ja	Ja	Nein

Tabelle 6.1: Wiederverwendung existierender Vokabulare und Ontologien, sowie einzelner Ressourcen daraus.

Klassen- und Eigenschaften sowie Inverse-Eigenschaften und Einschränkungen in Form von Domain und Range. Weitere Restriktionen werden nicht beachtet und fließen folglich bei der Wiederverwendung nicht ein.

Protégé

Protégé agiert wesentlich flexibler, da es Dateien von der Festplatte und aus dem Internet lesen kann. Das Programm verarbeitet nahezu jedes valide Vokabular oder Ontologie, bei denen Hierarchien, Verknüpfungen und selbst Restriktionen keine Hürde darstellen. Außerdem werden zudem mögliche Instanzen aus den Klassen und Eigenschaften geladen, welche Schlussfolgerungen ermöglichen. Möchte der Benutzer eine neue Ontologie erstellen, welche eine fremde Ontologie verwendet oder auf dieser basiert, kann Protégé die Ontologie importieren und diese z. B. anschließend der automatischen Vervollständigung zur Verfügung stellen. Folglich stellt Protégé den umfassendsten Import dar.

Website-Ontologie

Dem gegenüber weißt die Website-Ontologie in diesem Bereich ein großes Defizit auf und sollte um die Verwendung fremder Vokabulare und Ontologien erweitert werden. Für den Import kann auf das Evoc-Modul aus dem RDFx-Paket zurückgegriffen werden. Anschließend können dem Benutzer beim Anlegen von Verknüpfungen durch die automatische Vervollständigung fremde Begriffe vorgeschlagen werden. Bei der Ausgabe müssen in weiterer Folge die verwendeten Vokabulare und Ontologien in den Kopf der Website-Ontologie eingebunden werden.

6.2 Modifikation der eigenen Ontologie

Jede Ontologie wird im Laufe des Entwicklungszyklus (s. Abschnitt 4.2) mehrfach modifiziert. Das umfasst zum einen das Hinzufügen, Bearbeiten und Löschen von Ressourcen und zum anderen das Verknüpfen von einer oder

	RDFx v6 / v7	*Protégé*	*Siteonto*
Klassen und Eigenschaften	Teilweise / Nein	Ja	Ja
Hierarchien	Nein / Nein	Ja	Ja
Verknüpfungen	Teilweise / Nein	Ja	Teilweise
Restriktionen	Teilweise / Nein	Ja	Nein
Instanzen	Ja / Ja	Ja	Ja

Tabelle 6.2: Aktionen zur Modifikation der Ontologie.

mehrerer Ressourcen, das Bilden von Hierarchien und das Hinzufügen von Restriktionen. Diese Anforderungen sollte ein Ontologie-Editor auf möglichst unkomplizierte und benutzerfreundliche Weise erfüllen.

RDFx-Modul

Die Tabelle 6.2 stellt die eben angeführten Aktionen für die verschiedenen Programme dar. Die RDFx-Module müssen dazu differenziert nach Versionen betrachtet werden, da sie sich im Funktionsumfang start unterscheiden. In der Version für Drupal 6 ist es möglich ein Site-Vokabular aus den Inhaltstypen zu erzeugen und diese wiederum mit den erstellten Klassen und Eigenschaften zu annotieren. Ein manueller Eingriff in das Vokabular ist nur über den Umweg der Inhaltstypen möglich, wodurch eine Modifikation von Klassen und Eigenschaften nur teilweise erfüllt wird. Auch eine Verknüpfung zu anderen Ressourcen ist lediglich teilweise gegeben, da der Domain-Range-Wertebereich nur mit eigenen Ressourcen gefüllt wird. Fremde Ressourcen können weder bei der Bildung von Verknüpfungen, noch für Hierarchie genutzt werden. Einziger Ausweg ist es die Instanzen zusätzlich mit einer fremden Ressource zu annotieren. Restriktionen werden aus den Feldern der Inhaltstypen ausgelesen und in Kardinalitäten umgewandelt [10]. Mit der Version für Drupal 7 wandelt sich das Bild drastisch. Das Site-Vokabular ist bis zur Fertigstellung der Arbeit nicht portiert, wodurch die Mehrheit der Aktionen entfallen. Lediglich das Instanziieren von fremden Ressourcen durch die Annotierung von Inhaltstypen ist erhalten geblieben.

Protégé

Protégé bietet eine vollständige Unterstützung für alle genannten Aktionen. Angefangen beim Hinzufügen, Bearbeiten und Löschen von Ressourcen, über das Bilden von Hierarchien bis zum Verknüpfen von eigenen mit importierten Ressourcen (siehe auch Abschnitt 6.1). Mit dem Ontologie-Editor können ebenso verschiedene Arten von Restriktionen, wie auch Instanzen von

Klassen und Eigenschaften erstellt werden. Der große Funktionsumfang geht allerdings mit einer vielschichtigen Benutzeroberfläche einher, welche eine steile Lernkurve für den Benutzer bedeutet.

Website-Ontologie

Die Website-Ontologie stellt einen größeren Funktionsumfang gegenüber den RDFx-Modulen zur Verfügung, kann mit Protégé jedoch nur in einigen Punkten mithalten. So lassen sich neben den konvertierten Inhaltstypen, benutzerdefinierte Klassen und Eigenschaften erstellen und zu Hierarchien anordnen. Verknüpfungen können lediglich in Form von Domain-Range-Beziehungen für Eigenschaften angegeben werden. Weitere Verknüpfungsarten sind ebenso wie Restriktionen derzeit nicht implementiert. Äquivalent zu den RDFx-Modulen können Instanzen durch Annotieren der Inhaltstypen mit den Klassen und Eigenschaften erstellt werden. Dazu greift die Website-Ontologie allerdings auf die bestehende RDFx-Funktionalität zurück.

Die gesamte Benutzeroberfläche der Website-Ontologie vermittelt einen einheitlichen Eindruck, da sie sich in den Administrationsbereich von Drupal integriert. Die Drupal-Benutzeroberfläche wurde mit der Version 7 von Grund auf erneuert und benutzerfreundlicher gestaltet. Davon profitiert natürlich auch das Website-Ontologie-Modul, welches die Drupal API für Tabellen, Formulare usw. verwendet. Jedoch lässt sich das Modul auch in diesem Rahmen weiter verbessern. So lassen sich die Klassen und Eigenschaften nicht wie bisher in einer Ansicht, sondern in getrennten Ansichten auflisten und mit einer einfachen Suche ausstatten. Das fördert die Übersichtlichkeit und den Umgang mit vielen Ressourcen.

Beim Hinzufügen und Editieren von Ressourcen können das Label und die Kommentare bisher nur in einer einzigen Sprache angelegt werden, welche standardmäßig der Sprache des Administrationsbereiches entspricht. Soll die Beschreibung jedoch in verschiedenen Sprachen angelegt werden, so muss der Benutzer weitere Felder für jede Sprache hinzufügen können. Die Erweiterung impliziert, dass der Maschinenname des Begriffs unter Umständen nicht mehr einfach aus dem Label generiert werden kann.

Ein weiterer Punkt betrifft die eingeschränkte Auswahl von Eigenschaften. Bisher existieren die `owl:ObjectProperty` und `owl:DatatypeProperty`, welche bereits automatisch unterschieden werden. Zudem wäre es wünschenswert, die Eigenschaften manuell als `owl:FunctionalProperty`, `owl:InverseProperty` usw. festlegen zu können. Damit kann die im Beispiel (s. Abschnitt 5.5.3) angesprochene Eigenschaft `site:isPartOf` korrekterweise als `owl:TransitiveProperty` deklariert werden. Die Verwendung der Eigenschaft erfordert ebenfalls eine Erweiterung des Eingabeformulars für Klassen. Das Programm 6.1 stellt die zu erzeugende RDF/XML-Ausgabe dar, welche den Fachbereich mit der Eigenschaft `site:isPartOf` als Kindklasse der Universität einschränkt. Das bedeutet für das Formular, dass zu jeder Klasse

```
 1 <owl:Class rdf:about="#Department">
 2   <rdfs:subClassOf>
 3     <owl:Restriction>
 4       <owl:onProperty>
 5         <owl:TransitiveProperty rdf:ID="isPartOf"/>
 6       </owl:onProperty>
 7       <owl:hasValue rdf:resource="#University" />
 8     </owl:Restriction>
 9   </rdfs:subClassOf>
10 </owl:Class>
```

Programm 6.1: Mögliche Erweiterung des Moduls um individuelle Eigenschaften und Zuweisung zur Klasse.

individuelle Subklassen mit Restriktionen konfiguriert werden können. Die Herausforderung ist es, die Benutzeroberfläche möglichst einfach zu gestalten und ein Datenbankschema zu entwickeln, welches sich flexibel an die Gegebenheiten anpasst.

Neben den bereits angeführten Verbesserungsvorschlägen, stellt die Umstellung der Persistierung auf die RDFx-Tabellen eine der tiefgreifendsten Änderungen dar. Die Daten würden dann komplett in den RDFx-Tabellen abgelegt werden und RDFx direkt zur Verfügung stehen. Damit entfällt das ständige Löschen und Importieren der Website-Ontologie (s. Abschitt 5.4.7), wodurch die Datenbank entlastet wird. Der Wechsel ergibt weiters den Vorteil, dass das Datenbankmodell von RDFx flexibler ist als das der bisherigen Website-Ontologie. Für die angesprochene Erweiterung um individuelle Eigenschaften und Restriktionen werden jedoch weitere Tabellen benötigt, da diese Art von Daten im bisherigen RDFx-Schema nicht vorgesehen ist.

6.3 Veröffentlichung und Verwendung

Nach der angesprochenen Erstellung und Modifikation der Ontologie folgt die Verwendung der veröffentlichen Ontologie. Eine Veröffentlichung ist damit nicht nur die Grundlage für die Verwendung auf der eigenen Seite, sondern fördert auch die Wiederverwendung der Ontologie im Allgemeinen und folgt damit dem Gedanken des Semantic Webs. Die Anforderung an die Programme ist, dass sich die Ontologie im Internet für den Benutzer auf einfache Art und Weise veröffentlichen und auf der eigenen Seite verwenden lässt.

RDFx-Modul

Die Tabelle 6.3 stellt die Aktionen zur Veröffentlichung und anschließenden Verwendung der Ontologie dar. Die RDFx-Module publizieren in der Version für Drupal 6 das generierte Site-Vokabular unter einer definierten URL.

	RDFx v6 / v7	*Protégé*	*Siteonto*
Veröffentlichung im Internet	Ja / Nein	Nein	Ja
Annotierung von Website-Inhalten	Ja / Ja	Nein	Teilweise

Tabelle 6.3: Aktionen zur Veröffentlichung und Verwendung der Ontologie.

Mit den erstellten Klassen und Eigenschaften werden automatisch die jeweiligen Inhaltstypen annotiert. Des Weiteren lassen sich externe Ontologien importieren und ebenfalls zuweisen. Mit der Version 7 bleibt lediglich die Funktion zum Importieren und Annotieren von Inhaltstypen erhalten. Die Funktionalität des Site-Vokabulars ist (derzeit) nicht portiert.

Protégé

Protégé bietet im Bereich der Veröffentlichung und Verwendung, gegenüber den anderen genannten Programmen, keine entsprechende Funktionalität an. Die Ontologien lassen sich in verschiedenen Formaten als Datei abspeichern und müssen anschließend manuell über einen Server im Internet bereitgestellt werden. Bei möglichen Modifikationen der Ontologie müssen die Schritte erneut ausgeführt werden. Ebenso verhält es sich mit der Verwendung der Ontologie, bei der die Inhalte einer Website mit Ressourcen annotiert wird. Dieser Vorgang muss vom Benutzer ebenfalls manuell durchgeführt werden, da Protégé keine Unterstützung anbietet.

Website-Ontologie

Das Modul veröffentlicht die Website-Ontologie unter einer definierten URI und aktualisiert diese bei Modifikationen automatisch. Für den Benutzer entsteht lediglich ein Aufwand durch das Annotieren der Inhaltstypen mit den einzelnen Ressourcen.

Eine Verbesserung könnte daher die automatische Zuweisung der konvertierten Klassen und Eigenschaften zu den jeweiligen Inhaltstypen ergeben. Damit würde eine nachträgliche manuelle Zuweisung überflüssig werden. Nicht zuletzt wäre es wünschenswert, wenn Namensänderungen der Ressourcen nachverfolgt und automatisch in den zugewiesenen Inhaltstypen angepasst werden könnten.

6.4 Fazit

In den vorangegangenen Abschnitten wurden durch verschiedene Vergleiche zwischen den RDFx-Modulen, dem Ontologie-Editor Protégé und dem Website-Ontologie-Modul die unterschiedlichen Stärken und Schwächen der Programme herausgearbeitet. Protégé eignet sich für die umfassende Entwicklung von Ontologien und bietet im Rahmen des Vergleiches den größten Funktionsumfang an. An die Grenzen stößt der Editor bei der Veröffentlichung der erstellten Ontologie, da diese vom Benutzer manuell veröffentlicht und auf Website-Inhalte angewendet werden muss.

Die RDFx-Module müssen nach Versionen differenziert betrachtet werden. Lediglich die Version für Drupal 6 generiert aus den Inhaltstypen ein Site-Vokabulars und annotiert diese wiederum mit den Ressourcen. Eine Erweiterung um benutzerdefinierte Ressourcen ist jedoch nicht möglich. Das Vokabular wird ohne Eingreifen des Benutzers automatisch unter einer definierten URI abgelegt. Beide Modul-Versionen (für Drupal 6 und 7) bieten eine sehr gute Unterstützung beim Import von externen Vokabularen und Ontologien und können diese ebenfalls den Inhaltstypen zuweisen.

Der Funktionsumfang der Website-Ontologie ist mit dem der RDFx-Module für Drupal 6 vergleichbar. Einer der größten Unterschiede und zugleich Vorteile sind die Modifikation der konvertierten Inhaltstypen sowie die Möglichkeit benutzerdefinierte Ressourcen zur Ontologie hinzuzufügen. Die bestehende, prototypische Anbindung des implementierten Moduls an die RDFx-Module hat sich bereits für die Annotation der Inhaltstypen als nützlich erwiesen. Jedoch weißt die Implementierung derzeit ein großes Defizit im Bereich des Imports und der Verwendung von externen Ontologien auf. Dieser Mangel sollte bei einer weiteren Entwicklung berücksichtigt und kann durch eine tiefere Integration der RDFx-Module behoben werden. Entsprechende Vorschläge wurden in den vorangegangenen Abschnitten beschrieben.

Kapitel 7

Zusammenfassung

Website-Ontologien stellen eine besondere Form der Ontologien dar, da sie thematisch eng mit der Website verknüpft sind und das gesamte Wissen oder einen Aspekt davon in semantischer Form aufbereiten. Sie basieren auf den standardisierten Sprachen des Semantic Webs RDF, RDFS und OWL, die sowohl von Menschen als auch von Maschinen gelesen und verstanden werden.

Der Einsatz eines Content Management Systems erleichtert die Erstellung einer Website-Ontologie. Deshalb wurden die bekanntesten Open-Source CMS hinsichtlich ihrer semantischen Unterstützung analysiert. Joomla! und TYPO3 bieten bisher keine oder nur sehr wenige Funktionalitäten, welche das Semantic Web fördern. Im Gegensatz dazu stehen für die Systeme WordPress und Drupal eine ganze Reihe von semantischen Modulen und Erweiterungen zur Verfügung.

An die Marktanalyse anschließend, wurden die acht Schritte nach Noy und McGuiness [20] zur Entwicklung einer Ontologie betrachtet. Davon abgeleitet wurde der Entwicklungszyklus der Website-Ontologie vorgestellt. Die Ontologie bezieht sich auf das Themengebiet der zugehörigen Internetpräsenz.

Die praktische Umsetzung der Arbeit stützt sich auf das CMS Drupal, welchem bereits in der Marktanalyse eine gute semantische Unterstützung attestiert wurde. Aus der Veröffentlichung von Corlosquet et al. [10] und den zugehörigen RDFx-Modulen wurde der Ansatz für die Entwicklung der Website-Ontologie abgeleitet. Das implementierte Modul wandelt die Inhaltstypen und deren Felder von Drupal in semantische Klassen und Eigenschaften um. Zusätzlich kann der Benutzer weitere Ressourcen manuell hinzufügen und alle Elemente untereinander verknüpfen. Die erstellte Website-Ontologie wird unter einer vordefinierten URI bereitgestellt. Mit dem Import der Ontologie schließt sich der Kreis, da durch die bestehenden RDFx-Module die Klassen und Eigenschaften wiederum den Inhaltstypen und Feldern zugewiesen werden können. In einem Szenario wurde das entwickelte Modul anhand

einer Beispielseite getestet und deren Umfang und Handhabung im Arbeitsablauf bewertet.

Im letzten Teil der Arbeit wurde die Implementierung mit den RDFx-Modulen, sowie dem Ontologie-Editor verglichen. Letzteres ist für die Entwicklung einer umfassenden Ontologie unerlässlich, besitzt jedoch Defizite bei der Veröffentlichung und Nutzung. Die Website-Ontologie ist z. B. bei der Modifikation der Ontologie den RDFx-Modulen überlegen, allerdings beim Import von fremden Ontologien benachteiligt. Dieser Mangel und andere Funktionen können durch eine engere Verzahnung der beiden Module behoben bzw. erweitert werden.

Ungeachtet der semi-automatischen Unterstützung bei der Erstellung wird auch weiterhin ein Mehraufwand bei der Entwicklung einer Website-Ontologie für den Benutzer anfallen. Folglich wird der Ansatz nicht in den Massenmarkt vordringen, sondern eine Nische des Semantic Webs besetzen. Nichtsdestotrotz fügt auch die Website-Ontologie ein weiteres Teil zum Semantic Web hinzu und hilft so, die Vision einer frei verfügbaren Wissensdatenbank zu verwirklichen.

Anhang A

Inhalt der CD-ROM

Der Inhalt der CD-ROM ist wie in der folgenden Auflistung zu sehen aufgebaut. Im Verzeichnis *code* befindet sich der Quellcode für das Website-Ontologie-Modul. Das Verzeichnis *szenario* enthält die Drupal-Installation aus dem Abschnitt 5.5. Die referenzierten Onlinequellen befinden sich im Verzeichnis *onlinequellen*. Im Ordner *thesis* liegt diese Arbeit in elektronischer Form.

Pfad: /code

siteonto.tar.gz	Installationsdatei und Quellcode des implementierten Website-Ontologie-Moduls

Pfad: /onlinequellen

*.pdf	Genutzte Onlinequellen im PDF-Format

Pfad: /szenario

drupal-installation.zip .	Konfigurierte Drupal-Installation mit alle Modulen, Inhaltstypen, usw.
datenbank.mysql.gz . .	Export der MySQL-Datenbank mit dem Drupal-Modul *Backup and Migrate*[1]
readme.txt	Hinweise zur Installation

Pfad: /thesis

Stitz_Holger_2011.pdf	Diese Arbeit im PDF-Format

[1] http://drupal.org/project/backup_migrate

Literaturverzeichnis

[1] Antoniou, G. und F. van Harmelen: *A semantic Web primer (Cooperative information systems series)*. The MIT Press, Cambridge, USA, 2. Aufl., 2008.

[2] Bechhofer, S., F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider und L. A. Stein: *OWL Web Ontology Language Reference*, Feb. 2004. http://www.w3.org/TR/owl-ref/, Kopie auf CD-ROM (Datei onlinequellen/2004_w3c_owl-reference.pdf vom 13.09.2011).

[3] Beckett, D. und T. Berners-Lee: *Turtle - Terse RDF Triple Language*, März 2011. http://www.w3.org/TeamSubmission/turtle/, Kopie auf CD-ROM (Datei onlinequellen/2011_w3c_turtle.pdf vom 13.09.2011).

[4] Berners-Lee, T., J. Hendler und O. Lassila: *The semantic web*. Scientific American, 284:34–43, Mai 2001.

[5] Bratt, S.: *Semantic Web, and Other Technologies to Watch*, Jan. 2007. http://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb/0130-sb-W3CTechSemWeb.pdf, Kopie auf CD-ROM (Datei onlinequellen/2007_bratt_semantic-web.pdf vom 13.09.2011).

[6] Brickley, D. und R. Guha: *RDF Vocabulary Description Language 1.0: RDF Schema*, Feb. 2004. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/, Kopie auf CD-ROM (Datei onlinequellen/2004_w3c_rdf-schema.pdf vom 13.09.2011).

[7] Brickley, D. und L. Miller: *FOAF Vocabulary Specification*, Aug. 2010. http://xmlns.com/foaf/spec/, Kopie auf CD-ROM (Datei onlinequellen/2010_brickley_foaf-spec.pdf vom 13.09.2011).

[8] Candela, L., D. Castelli, N. Ferro, Y. Ioannidis, G. Koutrika, C. Meghini, P. Pagano, S. Ross, D. Soergel, M. Agosti und Others: *The DELOS Digital Library Reference Model. Foundations for Digital Libraries*. Techn. Ber. 507618, Institute of Information Science and Technologies (ISTI) and Italian National Research Council (CNR), Pisa, Italy, Dez. 2007.

[9] Cisco System Inc.: *Cisco Visual Networking Index: Forecast and Methodology, 2010 - 2015*, Juni 2011. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf, Kopie auf CD-ROM (Datei onlinequellen/2011_cisco_networking-index.pdf vom 13.09.2011).

[10] Corlosquet, S., R. Delbru, T. Clark, A. Polleres und S. Decker: *Produce and consume linked data with drupal!* In: *Proceedings of the 8th International Semantic Web Conference*, ISWC '09, S. 763–778, Berlin, Heidelberg, 2009. Springer-Verlag.

[11] Corlosquet, S., A. Polleres, R. Cyganiak und S. Decker: *Semantic web publishing with drupal.* Techn. Ber. DERI-TR-2009-04-30, DERI - Digital Enterprise Research Institute, Galway, Ireland, Apr. 2009. http://www.deri.ie/fileadmin/documents/DERI-TR-2009-04-30.pdf.

[12] Gams, E. und D. Mitterdorfer: *Semantische Content Management Systeme.* In: Blumauer, A. und T. Pellegrini (Hrsg.): *Social Semantic Web: Web 2.0 - Was nun?*, Kap. 11, S. 207–226. Springer-Verlag, Berlin Heidelberg, Deutschland, 1. Aufl., 2008.

[13] Hitzler, P., M. Krötzsch, S. Rudolph und Y. Sure: *Semantic Web.* Springer-Verlag, Berlin / Heidelberg, Deutschland, 2008.

[14] Kampffmeyer, U.: *Enterprise Content Management: Die unternehmensweite Informationsplattform der Zukunft*, Juni 2003. http://www.project-consult.net/Files/IXOS_ECM_20030624.pdf, Kopie auf CD-ROM (Datei onlinequellen/2003_kampffmeyer_ecm.pdf vom 13.09.2011).

[15] Kempkens, A. und H. Graf: *Das Joomla!-Entwicklerhandbuch.* Addison-Wesley, München, Deutschland, 2005.

[16] Kinner, J. A.: *DSpace History System: RDF Schema Design*, Mai 2003. http://simile.mit.edu/reports/dspace-history/design.pdf, Kopie auf CD-ROM (Datei onlinequellen/2003_kinner_dspace.pdf vom 13.09.2011).

[17] Kruk, S., S. Decker und L. Zieborak: *Jeromedl - adding semantic web technologies to digital libraries.* In: Andersen, K., J. Debenham und R. Wagner (Hrsg.): *Database and Expert Systems Applications*, Bd. 3588 d. Reihe *Lecture Notes in Computer Science*, S. 716–725. Springer-Verlag, Berlin / Heidelberg, Deutschland, 2005.

[18] Lohr, J. und A. Deppe: *Der CMS-Guide. Content Management-Systeme: Erfolgsfaktoren, Geschäftsmodelle, Produktübersicht.* Vieweg, Braunschweig, Deutschland, 2001.

[19] Möller, K., U. Bojars und J. Breslin: *Using semantics to enhance the blogging experience*. In: Sure, Y. und J. Domingue (Hrsg.): *The Semantic Web: Research and Applications*, Bd. 4011 d. Reihe *Lecture Notes in Computer Science*, S. 679–696. Springer-Verlag, Berlin / Heidelberg, Deutschland, 2006.

[20] Noy, N. F. und D. L. McGuinness: *Ontology development 101: A guide to creating your first ontology*. Techn. Ber. KSL-01-05, Stanford Knowledge Systems Laboratory, Stanford, CA, USA, März 2001. http://www.ksl.stanford.edu/KSL_Abstracts/KSL-01-05.html.

[21] OWL Working Group: *OWL 2 Web Ontology Language Document Overview*, Okt. 2009. http://www.w3.org/TR/owl2-overview/, Kopie auf CD-ROM (Datei onlinequellen/2009_w3c_owl2.pdf vom 13.09.2011).

[22] Phuoc, D. L. und N. A. Rakhmawati: *Showcases of light-weight RDF syndication in Joomla*, Juni 2008. http://triplify.org/Challenge/Nominations/files?get=lephuoc_joomla.pdf, Kopie auf CD-ROM (Datei onlinequellen/2008_lephuoc_joomla.pdf vom 19.09.2011).

[23] Pouchard, L., N. Ivezic und C. Schlenoff: *Ontology engineering for distributed collaboration in manufacturing*. In: *Proceedings of the AIS2000 Conference*, Bd. 129, S. 2865. Tuscon, AR, USA, März 2000.

[24] Prud'hommeaux, E. und A. Seaborne: *SPARQL Query Language for RDF*, Jan. 2008. http://www.w3.org/TR/rdf-sparql-query/, Kopie auf CD-ROM (Datei onlinequellen/2008_w3c_sparql.pdf vom 13.09.2011).

[25] Shreves, R.: *Open Source CMS Market Share Report*, Dez. 2010. http://www.waterandstone.com/sites/default/files/2010%20OSCMS%20Report.pdf, Kopie auf CD-ROM (Datei onlinequellen/2010_shreves_os-cms-market-share.pdf vom 13.09.2011).

[26] Smith, M. K., C. Welty und D. L. McGuinness: *OWL Web Ontology Language Guide*, Feb. 2004. http://www.w3.org/TR/owl-guide/, Kopie auf CD-ROM (Datei onlinequellen/2004_w3c_owl-guide.pdf vom 13.09.2011).

[27] Synak, M., S. R. Kruk und K. Zimmermann: *Marcont initiative - the ontology for the librarian world*. In: *Demosession of the 9th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2005)*. Wien, Österreich, Sep. 2005.

[28] van 't Ende, B.: *TYPO3 presentation CeBIT 2010*. Hannover, Deutschland, März 2010. http://benvantende.blogspot.com/2010/05/typo3-presentation-cebit-2010.html, Kopie auf CD-ROM (Datei onlinequellen/2010_van 't ende_typo3-presentation.pdf vom 13.09.2011).

[29] VanDyk, J. K. und D. Buytaert: *Pro Drupal Development, Second Edition*. Apress, Berkely, CA, USA, 2. Aufl., 2008.

[30] W3Techs: *Market share trends for content management systems*, Sep. 2011. http://w3techs.com/technologies/history_overview/content_management, Kopie auf CD-ROM (Datei onlinequellen/2011_w3techs_cms-market-share.pdf vom 13.09.2011).

[31] Wahl, H., M. Linder und A. Mense: *RDFCreator - A Typo3 Add-On for Semantic Web based E-Commerce*. In: *Proceedings of I-SEMANTICS '08*, S. 186–189, Graz, Österreich, Sep. 2008.