User-orientiertes Interface zum Anreichern von Redaktionstexten mit semantischen Informationen in Web-Content-Management-Systemen

Stefan Bauer

DIPLOMARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im Juli 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 21. Juni 2011

Stefan Bauer

Inhaltsverzeichnis

Vorwort

Diese Diplomarbeit ist zum Abschluss meines Masterstudiums entstanden und obwohl man sich nach Vollendigung eines solch arbeitsaufwendigen Projektes am liebsten selbst auf die Schultern klopft, gibt es doch viele Menschen, die im Laufe der Jahre, aber auch speziell in den letzten Monaten durch ihre Unterstützung dazu beigetragen haben, dass diese Arbeit überhaupt geschreiben werden konnte.

Zuerst möchte ich mich bei meiner Mutter, Rosemarie Bauer, bedanken, die 2003 verstorben ist, bis dahin aber immer für mich da war. Genau so möchte ich mich bei meiner Familie, im speziellen Ernst und Petra Bauer, bedanken, ohne deren Beistand ich mein Studium wohl nicht geschafft hätte.

Ebenfalls hervorzuheben ist die Unterstützung meiner Freunde, Studien- und Arbeitskollegen sowie im besonderen die unendlichen Stunden, die Kerstin Dugall, Fabian Böhme und Andreas Lamann damit verbracht haben, diese Diplomarbeit immer und immer wieder Korrektur zu lesen.

Abschließend möchte ich mich noch bei Mag. DI Dr. Andreas Stöckl für seine kompetente und freundliche Betreuung mit stets raschen und verlässlichen Antworten bedanken.

Kurzfassung

Diese Diplomarbeit beschäftigt sich mit semantischer Informationsaufbereitung im Internet. Sie legt den Fokus dabei auf Verständnis- und Interaktionsmöglichkeiten für normale Webredakteure, die bisher kaum Kontakt mit dieser Thematik hatten.

Ergänzend dazu wurde der *Semantic Object Editor* für das Content Management System *Typo3* entwickelt. Diese Erweiterung bietet Redakteuren die Möglichkeit, Texte einerseits mit ihnen bereits vertrauten Schritten zu verwalten und andererseits durch erweiterte Funktionalitäten diese Inhalte mit semantischen Annotationen anzureichern.

Der Zugang zu Semantic Web-Technologien war im letzten Jahrzehnt durch sehr komplexe Systeme meist nur für Experten möglich. Trotz des enormen Wachstums der Linked-Data-Cloud fehlten Anwendungen, die normalen Webusern den Einstieg ins Semantic Web erlaubten. Zentrales Ziel dieser Diplomarbeit ist daher die Konzeption und die prototypische Umsetzung eines Interfaces, das normalen Webredakteuren erlaubt, ohne Vorwissen am Semantic Web teilzunehmen.

Es werden grundlegende Informationen wie die Idee des Semantic Web sowie die technologischen Bausteine dargelegt und mit praktischen Beispielen illustriert. Existierende Ansätze zur semantischen Anreicherung von Texten sowie wegweisende Projekte und Ideen werden vorgestellt. Diese werden zusammen mit dem *Semantic Object Editor* als auch den offenen Problemen von userzentrierten Interfaces und von Semantic Web kritisch besprochen.

Abstract

This thesis focuses on semantic information processing in the World Wide Web. It addresses the capability of normal web users without proper experience to understand and interact with given Semantic Web applications.

The thesis project, an extension for the CMS Typo3 called *Semantic Object Editor*, is also presented. It enables users to manage their content with familiar steps of the procedure while adding new functionality to semantically annotate given texts.

In spite of the tremendous growth of the linked-data cloud during the last decade, there were hardly any applications for normal web users to engage in the Semantic Web due to the fact that the technology was primarily targeted at experts. Hence, the central goal of this thesis is to design and develop a prototypical interface which enables users without any experience to be part of the Semantic Web.

It presents a comprehensive overview of the Semantic Web and illustrates the technological backbone with useful examples. The work also showcases existing projects as well as groundbreaking approaches and ideas of dealing with semantic annotation. It critically discusses these and the *Semantic Object Editor* in addition to the challenges facing semantic user-interfaces as well as the open questions regarding the Semantic Web.

Kapitel 1

Einleitung

1.1 Motivation

Der Grundansatz des Semantic Web lässt sich schnell erklären: im World-Wide-Web existiert mittlerweile eine enorme Datenmenge, die nur unter einer sehr groben Struktur aufbereitet wird, in kaum einer Beziehung zueinander steht und auch von Maschinen inhaltlich kaum verarbeitet werden kann.

Somit ist man inzwischen mit der Tatsache konfrontiert, eine erhebliche Menge an Wissen angesammelt zu haben, welches allerdings nur sehr schwer zugänglich ist und auch durch gängige Suchmaschinen, die mehrheitlich mit Volltext-Suchalgorithmen arbeiten, noch nicht vollständig und vor allem sinngemäß erfasst und verarbeitet werden kann.

In diesem Zusammenhang bieten Content-Management-Systeme eine einfache und komfortable Möglichkeit, Webseiten ohne Programmierkenntnisse zu verwalten und Inhalte zu veröffentlichen. Genau dies soll nun genutzt werden, damit Redakteure Inhalte mit semantischen Informationen anreichern können und um das Internet von einem *Web of Documents* zu einem *Web of Data* zu machen.

1.2 Zielsetzung

Ziel dieser Diplomarbeit ist es, bestehende Möglichkeiten der semantischen Wissensaufbereitung im Web zu untersuchen, etwaige Vor- und Nachteile herauszuarbeiten und Rückschlüsse darauf zu ziehen, warum sich Semantic Web trotz dieser Möglichkeiten auf dem Markt noch nicht erfolgreich etablieren konnte. Der Fokus soll dabei klar auf die Verbindung zwischen semantischen Annotationen und Content-Management-Systemen gelegt werden, da sich diese als Administrationstools für normale Redakteure durchgesetzt haben und damit ein sicheres, akzeptiertes und strukturiertes Trittbrett für weitere Entwicklungen des Semantic Web bieten.

Kernstück der Arbeit werden dabei Wissensaufbereitung und Annotati-

onsmöglichkeiten der jeweiligen Systeme sein, und wie normale Webuser die Möglichkeit haben, redaktionelle Texte und Inhalte mit semantischen Annotationen, Klassen und Objekten aufzubereiten.

Ergänzend wurde im Rahmen des Thesis Projekts eine Schnittstelle für das Content-Management-System *Typo3* entwickelt. Diese Extension trägt den Titel *Semantic Object Editor* und bietet Redakteuren eine schnelle und leicht verständliche Möglichkeit, erstellte Texte mit semantischen Informationen anzureichern.

1.3 Gliederung der Arbeit

Die vorliegende Diplomarbeit gliedert sich in acht Kapitel. Nach der Einführung in Kapitel 1 werden in Kapitel 2 die Grundlagen des Semantic Web behandelt.

Kapitel 3 beschäftigt sich mit den verwandten Arbeiten und legt dar, welche Ansätze und Projekte es in diesem speziellen Bereich des Semantic Web bisher gibt, legt vorhandene Vor- und Nachteile offen und zeigt neue sowie interessante Ansätze auf.

In Kapitel 4 wird analysiert, welche Anforderungen an eine semantische Schnittstelle in Typo3 gestellt werden und welche Funktionalitäten eine Erweiterung wie der *Semantic Object Editor* bieten sollte.

Darauf folgt Kapitel 5, in dem verschiedene Entwürfe für das Userinterface des *Semantic Object Editor* vorgestellt und besprochen werden.

Kapitel 6 bietet eine detaillierte Dokumentation des *Semantic Object Editor* und zeigt auf, wie die Erweiterung funktioniert, wie sie umgesetzt wurde und welche Technologien dabei zum Einsatz kamen.

Die Diskussion in Kapitel 7 widmet sich dann der Evaluierung des *Semantic Object Editor* und beleuchtet positive als auch negative Aspekte. Außerdem wird auch auf Erweiterungsansätzen und offenen Problemen des Semantic Web eingegangen.

In Kapitel 8 folgt die Schlussbemerkung als auch ein kurzer Ausblick, wie es mit dem Semantic Web weitergehen könnte.

Kapitel 2

Grundlagen des Semantic Web

Dieses Kapitel befasst sich kompakt mit den Grundlagen des Semantic Web und legt den nötigen Grundstock, der im Verlauf der Arbeit benötigt wird.

2.1 Definition

2.1.1 Die Grenzen des Web

Für viele Menschen ist das *World Wide Web* eine grenzenlose „Landschaft" an Möglichkeiten. Wenn man die rechtlichen Rahmenbedingungen erst einmal ausnimmt, bietet das Internet einen freien Platz, um sich selbst zu entfalten, sich zu bilden, soziale Kontakte zu knüpfen oder sich zu präsentieren.

Im vergangenen Jahrzehnt haben sich nicht nur die Möglichkeiten im Internet enorm gesteigert, auch die Zahl der User hat sich weltweit von 604 Millionen im Jahr 2002 auf 1501 Millionen im Jahr 2010 mehr als verdoppelt[1]. Damit stieg unweigerlich die Anzahl der erstellten Inhalte und damit die vorhandenen Daten. Dies führte dazu, dass mittlerweile eine für den Menschen unüberschaubare Menge an präsenten Informationen im Netz verhanden ist, deren struktureller Aufbau aber ausschließlich auf den Menschen als Endnutzer ausgerichtet ist - und genau hier liegt auch das Problem [19, S. 10].

Um dieses anschaulicher zu gestalten, nehmen wir den User Max Muster zu Hilfe, der auch in künftigen Beispielen immer wieder auftauchen wird. Max hegt schon lange den Traum, sich einen Jaguar zu kaufen. Er ist mit dem Internet einigermaßen vertraut und ist es gewohnt, das Internet nach Informationen für künftige Käufe zu durchsuchen. Wenn er nun in den heutigen Volltext-Suchmaschinen den Begriff *Jaguar* eingibt, wird er leider feststellen müssen, dass ihm nicht nur die neuesten Modelle der gewünschten Automarke geliefert werden, sondern auch viele Seiten gefunden werden, die sich

[1] Laut einer Studie des Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V. : http://www.bitkom.org/de/presse/49919_46069.aspx

mit dem Tier Jaguar beschäftigen. Warum ihm von der Suchmaschine beide Ergebnisse geliefert werden, ist leicht erklärt: die Suchmaschine kennt die genaue Wortbedeutung nicht und sucht einfach nach passenden Wortübereinstimmungen – die semantische Bedeutung des Wortes ist nicht bekannt.

Dieses Beispiel ist zwar trivial und einfach gestrickt, es wird jedoch ersichtlich, wo die Grenzen des sogenannten *normalen Webs* liegen, und wo Semantic Web ansetzt.

2.1.2 Semantic im World Wide Web

Die Idee des semantischen Web ist keinesfalls neu. Schon 1994 stellte Tim Berners-Lee, Erfinder der HTML und damit Mitbegründer des *World Wide Web*[2], auf der ersten *World Wide Web Conference* die Idee vor, dass Inhalte mit ihren semantischen Bedeutungen versehen werden sollten. Im Jahr 2001 beschrieb ein Artikel im *Scientific American* den Weg des World Web Web zum semantischen Web als eine Entwicklung von einer bloßen Dokumentensammlung für Menschen zu einer Informations- und Datensammlung, die von Computern verarbeitet werden können [30]. Dies hätte erstmal allerdings wenig mit dem strukturellen Aufbau des heutigen World Wide Web gemein und wäre in einigen Aspekten gar ein Rückschritt.

Eine reine Wissensdatenbank alleine, wenn auch mit allen zur Verfügung stehenden Informationen, kann sicher nicht der nächste Schritt in der Entwicklung des World Wide Web sein, denn das Internet besteht auch jetzt nicht nur aus bloßen inhaltlichen Dokumenten. Die sozialen und kommunikativen Aspekte, die speziell in den Zeiten des *Web 2.0* stark angewachsen sind, dürfen bei diesen Entwicklungen natürlich nicht einfach ignoriert werden. Pascal Hitzler beschreibt das Ziel des Semantic Web in seinem Buch *Semantic Web: Grundlagen* wie folgt [19, S. 12]:

> Finde Wege und Methoden, Informationen so zu repräsentieren,
> dass Maschinen damit in einer Art und Weise umgehen können,
> die aus menschlicher Sicht nützlich und sinnvoll erscheinen.

Um dies gewährleisten zu können, muss man für das Semantic Web Lösungen finden, semantische Informationen über die Bedeutung der Inhalte in bereits existierende Mechanismen einzubauen und neue Regeln schaffen, die auf bewährte Strukture des World Wide Web aufbauen und diese verbessern.

2.1.3 Motivation für Anwender

Die Motivation der Anwender war lange ein Stolperstein des semantischen Web. Da kaum semantische Daten im Umlauf waren, wurde von vielen Entwicklern kein Grund gesehen, passende Applikationen zu entwickeln, wo-

[2] http://de.wikipedia.org/wiki/Tim_Berners-Lee

durch Anwender wiederum sehr geringen Zugang zu semantischen Technologien hatten. Durch die großen Entwicklungsschritte während des letzten Jahrzehnts (siehe Kapitel 3) als auch dem Wachstum der Linked-Data-Cloud (siehe Abschnitt 2.4) ging das Semantic Web große Schritte auf den Anwender zu und wurde dadurch auch greifbarer.

Regierungen wie die US-Administration nutzen mittlerweile semantische Dienste, um die Transparenz in Regierungsdaten zu gewährleisten (siehe Abschnitt 3.3.2), während Google und Yahoo! angefangen haben, RDFa Informationen von Webseiten [29, S. xi] in die Suchalgorithmen zu integrieren.

Anwender können durch Linked-Data den Informationsgehalt anreichern und ihren Kunden damit eine gesteigerte Userexperience bieten, während sie im Hintergrund davon profitieren können, dass sie einerseits durch semantische Annotationen erreichen, in Suchmaschinen an den richtigen Stellen aufzutauchen, als auch andererseits ihre Daten für externe Dienste zur Verfügung stehen. Stellt also, zum Beispiel, ein Autohaus in der Nähe von Max Muster die Produktpalette in der Linked-Data-Cloud zur Verfügung und Max nutzt eine Applikation, die ihm die nächste Anlaufstelle für sein Traumauto ausgibt, profitiert davon nicht nur Max, sondern auch das Autohaus. Damit hilft das Semantic Web, Anbieter und Kunden besser und spezifischer aufeinander abzustimmen. In dem Artikel *A Semantic Future for AI* wird diese Ansicht geteilt: "'Semantic descriptions of desired products and services will enable better matching of buyers and sellers and enable dynamic coalitions of costumers [31]."'

Ein großes Anwendungsgebiet sind außerdem Wissensportale wie zum Beispiel DBpedia (siehe Abschnitt 2.4.2) oder BioPortal (siehe Abschnitt 3.3.3). Speziell hier lassen sich die Vorteile des Semantic Web mit den vernetzten Informationen und weitreichenden Schnittstellen voll auskosten und bieten Anwendern einen enormen Mehrwert. Genau dieser Mehrwert lässt sich dann auch auf kleinere Webseiten umlegen. Ob es nun ein Tierpark ist, der seinen Usern detaillierte Beschreibungen der Tierarten aus einer externen Quelle direkt auf der Webseite zur Verfügung stellt oder die Webseite eines Hotels, auf der die User direkt weitere Informationen zu den Attraktionen der Stadt geliefert bekommen - die Möglichkeiten sind gegeben. Es wird also Zeit, dass mehr Anwender damit in Kontakt kommen.

2.2 RDF

2.2.1 Allgemein

RDF steht für *Resource Description Framework*. Es wurde für die Beschreibung strukturierter Informationen entwickelt und offiziell von der W3C[3]

[3]Das World Wide Web Consortium (*W3C*) ist das Gremium zur Standardisierung von Webtechnologien: http://www.w3.org/

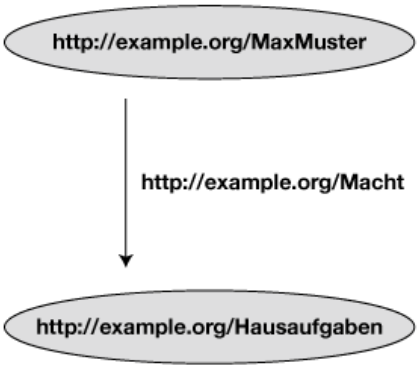


Abbildung 2.1: Ein einfacher RDF-Graph, der die Beziehung zwischen Max Muster und Hausaufgaben beschreibt.

empfohlen [25]. Im Gegensatz zu HTML und XML liegt nicht die korrekte Darstellung von Dokumenten, sondern die Kombination und Weiterverarbeitung der enthaltenen Informationen im Fokus [19]. Theoretisch ist man mit RDF in der Lage, jedes existierende Element bis ins kleinste Detail zu beschreiben [20]. Um Informationsabhängigkeiten ausreichend darstellen zu können, setzt RDF nicht wie XML auf eine Baum- sondern auf eine Graph-Struktur (siehe Abbildung 2.1). Grund dafür ist, dass bei zwei dargestellten Elementen, die durch eine Verbindung miteinander in Beziehung gesetzt wurden, kein Element dem anderen über- oder unterlegen ist und somit eine Baumstruktur nicht logisch wäre. Der strukturelle Aufbau des RDF-Graphen ist auch einer der Hauptfaktoren, warum semantische Daten in Form von Triples abgespeichert werden (siehe Abschnitt 2.3.2).

2.2.2 URI

Um Objekte in RDF eindeutig identifizierbar zu machen, werden diese mit sogenannten URIs (*Uniform Resource Identifiers*) ausgestattet. Bei einer URI handelt es sich um eine Oberklasse der aus dem Web schon bekannten URL (*Uniform Resource Locator*[4]). Dabei ist jede URL eine URI, da diese immer eindeutig ein Web-Dokument beschreiben. Eine URI ist aber nicht immer eine URL, da diese auch andere Objekte beschreiben können und daher im Web nicht ansprechbar sein müssen [19].

Die Idee dahinter ist einfach: Wenn ein Redakteur auf seiner Seite die Person *Max Muster* definieren möchte, muss er sichergehen können, dass *Max Muster* eindeutig bleibt und nicht mit anderen Personen mit dem gleichen

[4] http://de.wikipedia.org/wiki/Uniform_Resource_Locator

Namen, die jedoch auf anderen Seiten definiert wurden, verwechselt werden kann. Gewährleisten kann man dies durch eine eindeutige ID, in diesem Fall die URI, die sich aus dem zugewiesenen Namensraum der Seite als auch (meist) aus dem Objektnamen herleitet. Wie in Abbildung 2.1 zu sehen, ist in diesem Beispiel die URI von *Max Muster* http://example.org/MaxMuster.

Da es allerdings idealistisch ist, davon auszugehen, dass im Semantic Web über alle Datenquellen für jedes Objekt wirklich nur ein Bezeichner angelegt wird, gibt es sogenannte *URI aliases*, die festlegen, dass es sich bei zwei Elementen mit unterschiedlichen URIs um dasselbe Objekt handelt. Zum Beispiel definiert DBpedia[5] die deutsche Hauptstadt *Berlin* über die URI http://dbpedia.org/resource/Berlin, während Geonames[6] mit der URI http://sws.geonames.org/2950159/ arbeitet. Um dieses Problem zu beheben, sollte man bei diesen Objekten per *owl:sameAs*-Zuweisung klar definieren, dass es sich hierbei um identische Objekte handelt [7].

2.2.3 Stukturierung

Um Daten strukturiert auszugeben und die nötigen Verbindungen darstellen zu können, sind drei Elemente alleine jedoch nicht immer ausreichend. Daher bietet RDF die Möglichkeit sogenannter *Blank Nodes*. Diese dienen als Container-Knoten, die keine vollständige URI benötigen, nur durch eine interne ID zugewiesen werden können und es erlauben, Objekte mit mehreren Beziehungen zu modellieren.

Außerdem besteht die Möglichkeit, Objekte unter bestimmten Klassen und Unterklassen zusammenzufassen und Objekte mit Eigenschaften auszustatten. Da es sich hierbei nur um eine sehr grobe Darlegung der Strukturierungsmöglichkeiten handelt, sollte man sich bei Interesse näher mit der offiziellen Dokumentation von *W3C*[7] beschäftigen.

2.2.4 Serialisierung

Überblick

Durch die beschriebene Graph-Architektur von RDF-Elementen kann es jedoch bei der Erstellung einer serialisierten Repräsentation, wie zum Beispiel der Ausgabe als Dokument, zu erheblichen Komplikationen kommen. Problematisch ist vor allem, die komplexe Struktur so lesbar und verständlich wie möglich abzubilden [29, S. 68]. Einer der ersten Ansätze zur Ausgabe dieser Informationen war die Entwicklung des N3-Syntax[8], sowie die Darstellung als N-Triple. Dies war ein *lines-based*, *plain-text* Format[9]. Daraus

[5] http://www.dbpedia.org/
[6] http://www.geonames.org/
[7] http://www.w3.org/TR/rdf-primer/
[8] http://www.w3.org/2000/10/swap/Primer.html
[9] http://www.w3.org/TR/rdf-testcases/#ntriples

```
1 <?xml version='1.0'?>
2 <rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
3         xmlns:ex='http://www.example.org/'>
4   <rdf:Description rdf:about='http://www.example.org/MaxMuster'>
5     <ex:Macht rdf:resource='http://www.example.org/Hausaufgaben'/>
6   </rdf:Description>
7 </rdf:RDF>
```

Programm 2.1: RDF/XML Ausgabe.

folgte die Entwicklung der *Turtle*-Syntax, die zwar für Menschen eher leicht gelesen und verstanden werden kann, aber keine offizielle Empfehlung von *W3C* bekam[10].

RDF/XML

Bei RDF/XML handelt es sich um eine von der *W3C* empfohlene[25] Ausgabesyntax, die fälschlicherweise oft nur als RDF bezeichnet wird [29, S. 72]. Grundgedanke hinter der Entwicklung war die Tatsache, dass XML mittlerweile schon stark verbreitet ist und in fast jedes Framework eingelesen werden kann. Die unterschiedliche Strukturierung zwischen Baum- und Graphstruktur stellt hier auch kein Hindernis dar, da von XML nur die syntaktische Struktur vorgegeben wird [19, S. 42].

Die RDF/XML-Ausgabe des RDF-Elementes aus Abbildung 2.1 ist in Programm 2.1 zu sehen. Dabei dienen die Definitionen in Zeile 2 und 3 als sogenannte Präfixe. Diese dienen der Übersicht und können eingesetzt werden, um häufig auftretende Namensräume abzukürzen.

RDFa

Bei *RDFa* handelt es sich um keine reine RDF-Serialisierungssyntax, sondern um einen Weg, XHTML-Dokumente mit RDF-Daten anzureichern[29, S. 76]. Auch *RDFa* wird von der *W3C* empfohlen [1]. Der große Vorteil liegt dabei darin, dass die Inhalte nicht, wie zum Beispiel bei RDF/XML, getrennt, sondern in einem Dokument gesammelt veröffentlicht werden können. Vorbild dafür sind die sogenannten Microformate, die die Integration von neuen Technologien in ältere Formalismen ermöglichen [2].

Um RDF Elemente beschreiben zu können, werden die existierenden XHTML-Inhaltstags um XML Attribute wie unter anderem *about*, *rel*, *property*, *content* und *typeof* erweitert. Die genaue Bedeutung der einzelnen Tags kann in der offiziellen Dokumentation von W3C nachgelesen werden[11].

[10] http://www.w3.org/TeamSubmission/turtle/
[11] RDFa Attributes and Syntax http://www.w3.org/TR/rdfa-core/#s_syntax

```
1 <div xmlns="http://www.w3.org/1999/xhtml"
2     xmlns:ex='http://www.example.org/'>
3     xmlns:foaf="http://xmlns.com/foaf/0.1/">
4   <div about="http://www.example.org/MaxMuster">
5     <span property="foaf:name">Max Muster</span> macht
6     <span rel="ex:Macht" resource="http://www.example.org/Hausaufgaben">
      Hausaufgaben</span>
7   </div>
8 </div>
```

Programm 2.2: RDFa Ausgabe.

Die Annotationen können dabei bei jedem XHTML-Tag [29, S. 79] eingeführt werden.

In Programm 2.2 wurde wieder die Ausgabe von Abbildung 2.1 umgesetzt. Da in diesem Fall Max Muster auch als Name angeführt wird, macht es Sinn, das bekannte *foaf*-Vokabular (siehe Abschnitt 2.5) zu benutzen, um diesen auch als Namen zu definieren (siehe Zeile 5).

Da *RDFa* relativ leicht auf Webseiten einzubinden ist und die direkte Verbindung von Annotation und redaktionellem Text auch für Suchmaschinen sehr relevant ist, steigt die Anzahl der Webseiten, die RDFa einsetzen, rapide an [29, S. 79].

2.3 Datenaufbereitung

2.3.1 Allgemein

Eine der wichtigsten Grundlagen des Semantic Web sind die Daten, mit denen die Systeme arbeiten können. Essenziell ist daher auch, wie diese Daten im Hintergrund in Datenbanken abgespeichert werden. Die einfachste und wohl auch am stärksten verbreitete Art der Datenaufbereitung sind Tabellen [29, S. 6]. Ob nun per Hand oder über Software wie zum Beispiel *Microsoft Excel* - Tabellen sind einfach zu erstellen, zu bearbeiten und zu verstehen.

Die Datenverwaltung im Internet baut stark auf relationale Datenbanken und Content-Management-Systeme setzen dabei auf Produkte wie *MySQL*. Auch hier werden die Daten tabellarisch aufbereitet, können aber, wie der Name schon sagt, miteinander über bestimmte Schlüssel in Verbindung gesetzt werden. Der strukturelle Aufbau dieser Datenbanktabellen unterliegt aber nur sehr wenigen Konventionen und das Datenhandling obliegt dabei dem jeweiligen Programmierer. Siehe dazu als Beispiel Tab. 2.1.

Um mit diesen Beispieldaten über MySQL sinnvoll arbeiten zu können, wäre es nun natürlich sinnvoll, die Hobbies der einzelnen User aus dieser allgemeinen User-Tabelle zu entfernen und in eine externe Tabelle abzuspeichern und diese über Fremdschlüssel miteinander zu verbinden. Im Sinne

Tabelle 2.1: MySQL Datenbank-Tabelle.

Id	*Vorname*	*Alter*	*Hobbies*
1	Max	20	Schwimmen, Lesen, Laufen
2	Silke	23	Reiten, Lesen, Inline-Skating
3	Jane	16	Laufen, Musik, Fahrrad

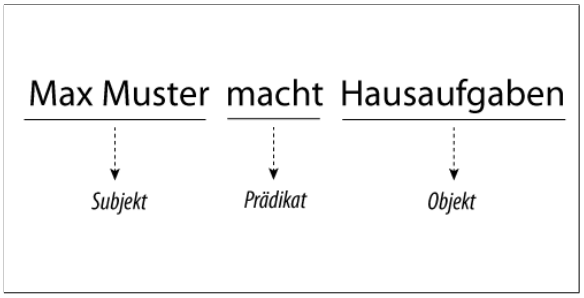


Abbildung 2.2: Beispiel für einen Datentriple.

der Einführung zu semantischer Datenaufbereitung sollte dies aber soweit genügen.

2.3.2 Daten-Triple

Diesem strukturellen Problem widmet sich das Konzept des sogenannten *Triple*. Hierbei handelt es sich um ein Konzept, das es erlaubt, semantische Daten weiterhin in relationalen Datenbanken abzulegen und das zusätzlich eine Brücke zum Semantic Web-Ausgabeformat RDF schlägt (siehe Abschnitt 2.2).

Der Grundgedanke hinter diesem Daten-Triple ist simpel: alle Daten werden in eine einzige Tabelle mit nur drei Feldern abgespeichert. Diese drei Felder leiten sich vom strukturellen RDF-Aufbau ab und man bezeichnet sie als *Subjekt*, *Prädikat* und *Objekt* [29, S. 19] - siehe als Beispiel Abbildung 2.2.

Erfahrene Programmierer werden bei diesem Ansatz vorerst einmal etwas verwundert sein, immerhin stellt er sich gegen gängige Konventionen im Bereich der Datenverarbeitung. Auf den ersten Blick öffnet man mit der Reduktion auf eine einzige Tabelle mit nur drei Feldern auch die Tür für viele mögliche Probleme wie redundante Daten und Verlust von Beziehungen. Man steht nun vor dem Problem, dass man mit nur drei Feldern alle Daten in der Datenbank repräsentieren muss. Wie man beim dem angeführten Beispiel (siehe 2.2) sieht, ist klar, dass Max seine Hausaufgaben macht. Um jetzt weitere Tätigkeiten oder Eigenschaften von Max zu beschreiben,

Tabelle 2.2: Datentriple Tabelle mit einfacher Beziehung.

Subjekt	*Prädikat*	*Objekt*
Max	Name	Max Muster
Max	Alter	20
Max	Beziehungsstatus	Ledig
Max	macht	Hausaufgaben

reicht es, ihn immer wieder als Subjekt anzuführen und bildet damit einen Schlüssel, über den die Beziehungen auch in einem Triple abbildbar werden.

Wie man nun in Tabelle 2.2 gut sehen kann, wird hier auch der Name als eigener Datentriple geführt. Wenn man sich die Grunddefinition der Triple noch einmal kurz in Erinnerung ruft, sollte auch klar sein, warum. Für einen menschlichen User ist es rein aus logischen Überlegungen, aus der Erfahrung und den grundsätzlichen Konventionen klar, dass es sich bei Max wohl um einen Namen handeln wird. Für den Computer ist es das natürlich nicht, da der Name nur als String angesehen wird, der alles bedeuten kann und in diesem Fall für einen Schlüssel, der die Eigenschaften verbindet. Daraus kann nun folgerichtig in diesem Daten-Triple *Max* auch durch eine Schlüsselvariable, wie zum Beispiel *P1*, ersetzt werden und es würde für die Maschine noch immer dieselbe Bedeutung haben.

Wenn man diesen Schlüssel nun als sogenannten Datenknoten sieht, der die zugehörigen Daten vereint und auch später auffindbar macht, wird auch schnell ersichtlich, wie man damit Beziehungen abbilden kann. Max ist mit dem Schlüssel P1 eindeutig idenfitiziert und kann damit auch bei anderen Triple als Objekt genutzt werden. Wenn man nun zum Beispiel alle Eigenschaften von Max und Silke Muster abspeichern möchte, zusätzlich aber eben auch noch definiert, dass die beiden Geschwister sind, ist dies nun mit den Triple-Knoten kein Problem mehr (siehe Tabelle 2.3). Mit diesen sogenannten Daten-Knoten ist es natürlich möglich, tiefere Beziehungen zu modellieren.

2.4 Linked Data

2.4.1 Allgemein

Wie in Abschnitt 2.1.2 besprochen wurde, ist eine der Idealvorstellungen des Semantic Web die Entwicklung von einem *Web der Dokumente* hin zu einem *Web der Daten*. Dieser Begriff mag noch etwas utopisch klingen, ist aber mittlerweile keine Zukunftsvision mehr. *Linked Data* wurde durch die großen Schritte, die im Bereich Semantic Web in den letzten zehn Jahren gemacht wurden, stark voran getrieben – wobei es auch den Standpunkt

Tabelle 2.3: Datentriple Tabelle mit Knoten.

Subjekt	*Prädikat*	*Objekt*
P1	Name	Max Muster
P1	Alter	20
P1	istBruderVon	P2
P2	Name	Max Muster
P2	Alter	23
P2	hatBruder	P1

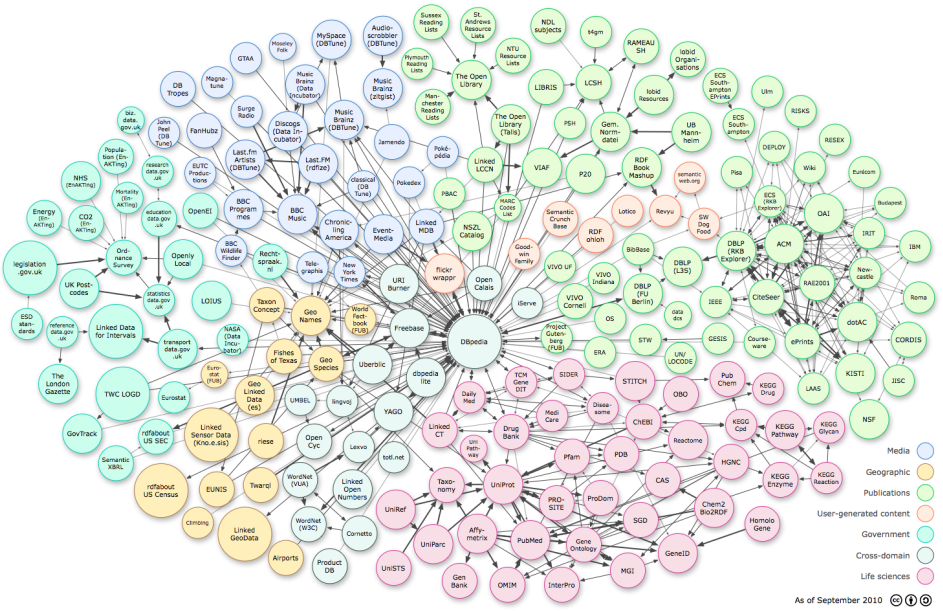


Abbildung 2.3: Linked-Data-Cloud Illustration von http://lod-cloud.net/.

gibt, dass erst der Durchbruch bei *Linked Data* den Dornröschenschlaf, in dem sich das Semantic Web in den letzten Jahrzehnten befand, beendet hat [15]. Die Anzahl der Datasets, die sich in der *Linked Data Cloud* befinden, hat sich alleine seit dem Jahr 2007 von 12 auf 203 fast verzwanzigfacht[8].

Aber was ist *Linked Data* überhaupt? Diese Frage lässt sich leicht beantworten: In den simpelsten Grundzügen handelt es sich dabei um Daten, die im Internet für Maschinen lesbar veröffentlich wurden. Dafür wurden sie spezifisch definiert und sind eindeutig identifizierbar. Sie verweisen durch angelegte Beziehungen auf andere Daten aus der *Linked Data Cloud* und werden im idealfall auch von anderen Datensammlungen referenziert [7].

2.4.2 DBpedia

Die wohl prominenteste Datenquelle zur Zeit ist DBpedia (siehe Abbildung 2.3). Dessen Datenbank beschreibt aktuell unter anderem mehr als 3.5 Millionen Elemente – darunter 364.000 Personen, 462.000 Orte, 99.000 Musikalben, 54.000 Filme, 17.000 Videospiele, 148.000 Organisationen, 169.000 Gattungen und 5.200 Krankheiten. Ihren Ursprung haben die Daten aus den Informationen der *Infoboxen* der normalen Wikipedia-Seiten, da diese stukturierte Informationen beinhalten[12].

2.5 Vokabular

Generell steht es im Semantic Web jedem Redakteur frei, bei der Objektzuweisung auf eigens definierte Klassen zurückzugreifen. Es ist allerdings *good practice*, bereits definiertes Vokabular aus öffentlichen Quellen zu verwenden, sollte es darunter schon passende Definitionen geben. Immerhin ist der Grundgedanke des Semantic Web, den Inhalt mit Annotationen anzureichern, damit dieser daraufhin von Maschinen auch sinnerfassend verarbeitet werden kann.

Wenn nun jede Webseite ihr eigenes Stammvokabular aufbauen würde, stünde man kurz darauf vor dem Problem, eine große Menge an neuen Annotationen in Texten vorzufinden, die jedoch für die Weiterverarbeitung in semantischen Applikationen nicht genutzt werden können, da sie hauptsächlich innerhalb des eigens definierten Vokabularraums gültig und verständlich sind.

Diesem Problem widmen sich zentrale Vokabelsammlungen, die es sich zum Ziel gesetzt haben, häufig benötigte Klassen und Eigenschaften vorzugeben und gesammelt zu veröffentlichen, um Redakteuren damit die Möglichkeit einer einheitlichen Annotierung zu geben. Erst durch diesen Ansatz ist es auch möglich geworden, die Linked-Data-Cloud aufzubauen. Es wäre zwar technisch umsetzbar, durch sogenannte *Alias*-Verweise auf andere Definitionen mit derselben Bedeutung zu zeigen; ohne eine zentrale Anlaufstelle wär aber auch dieser Ansatz nicht zielführend.

2.5.1 Friend of a Friend

Friend of a Friend ist unter der Abkürzung *foaf* [9] bekannt. Im Zentrum dieses Vokabulars stehen dabei Personen und Organisationen sowie Beziehungen, die diese untereinander und mit Dokumenten haben können. Da User ein zentrales Element im Web ausmachen, ist es nicht überraschend, dass *foaf* das zweithäufigste Vokabularset (siehe Abbildung 2.4) in der Linked-Data-Cloud ist.

[12] http://dbpedia.org/About

Foaf bietet dabei zum Beispiel die Definitionen für die Klassen *Agent*, *Organization*, *Group*, *Person*, *Document* und *Image*. Nennenswerte Beispiele für die zur Verfügung stehenden Attribute sind *firstName*, *familyName*, *age*, *phone* und *mbox*[13]. Das komplette Vokabular kann in der *foaf*-Spezifikation jederzeit nachgeschlagen werden[14].

Die Person *Max Muster* beschreibt man damit zum Beispiel wie folgt:

```
1 <foaf:Person rdf:about="#Max" xmlns:foaf="http://xmlns.com/foaf/0.1/">
2   <foaf:name>Max Muster</foaf:name>
3   <foaf:age>20</foaf:age>
4   <foaf:homepage rdf:resource="'http://www.example.org/MaxMuster" />
5   <foaf:openid rdf:resource="'http://www.example.org/MaxMuster" />
6   <foaf:img rdf:resource="/images/Max.jpg" />
7 </foaf:Person>
```

2.5.2 Good Relations

Bei *Good Relations* (gr) handelt es sich um ein Vokabular, das sich auf die semantische Annotierung von E-Commerce-Inhalten spezialisiert hat und seit 2005 konstant weiterentwickelt wird [17]. Die Grundidee dahinter ist, eine Brücke zwischen der semantischen und der kommerziellen Welt zu bauen und die strukturellen Vorteile, die im Semantic Web bisher verstärkt auf informationslastigen Diensten fokussiert waren, für die Wirtschaft zu nutzen.

In Abschnitt 2.1.3 wurde als Beispiel ein Autohaus vorgestellt, dass seine Produktpalette im Semantic Web veröffentlicht, um neue und bessere Wege zu finden, an kaufinteressierte Kunden zu kommen. Genau für diesen Ansatz sind Vokabelsammlungen von äußerster Wichtigkeit, da erst die konsistente Annotierung von Produkten externen Applikationen ermöglichen, die richtigen Rückschlüsse zu ziehen und dem User die passenden Ergebnisse zu liefern.

Good Relations bietet eine große Anzahl an Klassen, Eigenschaften und Beziehungen, die gezielt dafür entwickelt wurden, Produkte so detailliert wie möglich beschreiben zu können – *ActualProductOrServiceInstance*, *color*, *serialNumber*, *isConsumableFor* und *UnitPriceSpecification* sind dabei nur ein kleiner Auszug[18] [18].

Erwähnenswert ist dabei, dass die Sammlung schon einige große Befürworter für sich gewinnen konnte: so wird *gr* bereits von *Google*[15] und *Yahoo*[16] unterstützt, während *Bing*[17] die Unterstützung ebenfalls ankündigte. Firmen wie *Best Buy*, *Overstock.com*, *O'Reilly* und *Peek & Cloppenburg*[18] haben bereits damit begonnen, ihre Produktpalette damit anzureichern.

[13] Wird zum Beispiel für Email-Adressen benutzt.
[14] http://xmlns.com/foaf/spec/
[15] http://www.google.com
[16] http://www.yahoo.com
[17] http://www.bing.com
[18] http://wiki.goodrelations-vocabulary.org/References

Jan Myers, *Lead Web Development Engineer* bei *Best Buy*, sprach sich in einem Interview stark für die Entwicklung zum Semantic Web aus und erklärte, dass nicht nur die Kunden und der Verkauf davon profitieren, sondern auch die interne Datenverwaltung stark profitieren konnte [4].

Wie man in Abbildung 2.4 sehen kann, ist die Verbreitung des Vobabulars in der Linked-Data-Cloud noch eher gering. Es hat sich jedoch gerade in diesem Bereich in letzter Zeit enormes getan – so erschien die Google-Empfehlung zum Beispiel erst Ende 2010[19]). Im Weiteren baut das Diagramm hauptsächlich auf große Informationsdatenbanken, wodurch der geringe Anteil auch noch erklärbar wäre.

2.5.3 Weitere Beispiele

Neben *foaf* und *gr* gibt es noch eine große Menge an weiteren Vokabelsammlungen, die meist ein spezifiziertes Themengebiet behandeln und damit in ihrer Summe schon einen großen Bereich abdecken. Die folgenden Beispiele zeigen nur einige bekannte und stark verbreitete Exemplare auf und es besteht kein Anspruch auf Vollständigkeit. Bei Abbildung 2.4 ist die Verteilung in der Linked-Data-Cloud zu sehen.

Dublin Core

Dublin Core wird meist mit der Abkürzungen *dcterms* benutzt und ist zusammen mit dem Vorgänger (*dc*)[20] das am häufigsten referenzierte Vokabular. Es besteht aus 15 Klassen, die allesamt die Beschreibung von Dokumente unterstützen, wie zum Beispiel *creator*,*format*, *publisher* und *title*[21].

Simple Knowledge Organization System

Bei *Simple Knowledge Organization System* (skos) handelt es sich um ein Vokabular zur genaueren Beschreibung von Onthologien [21]. So existieren unter anderem Klassen und Eigenschaften wie *collection*, *concept*, *altLabel*, *changeNote*, *related* und *example* [22], die die normierte Darlegung von Fakten und Wissen vereinfachen sollen.

Semantically-Interlinked Online Communities

Semantically-Interlinked Online Communities (sioc) fokussiert sich auf die Darstellung von Informationen aus Online-Communities wie Foren, Wikis und Webblogs. Es bietet dafür unter anderem Klassen und Eigenschaften wie *community*, *user*, *post*, *topic*, *avatar* und *moderator_ of* [23].

[19] http://wiki.goodrelations-vocabulary.org/References
[20] http://dublincore.org/documents/dces/#DCTERMS
[21] http://purl.org/dc/elements/1.1/
[22] http://www.w3.org/TR/skos-reference/skos.html
[23] http://www.w3.org/Submission/2007/SUBM-sioc-spec-20070612/

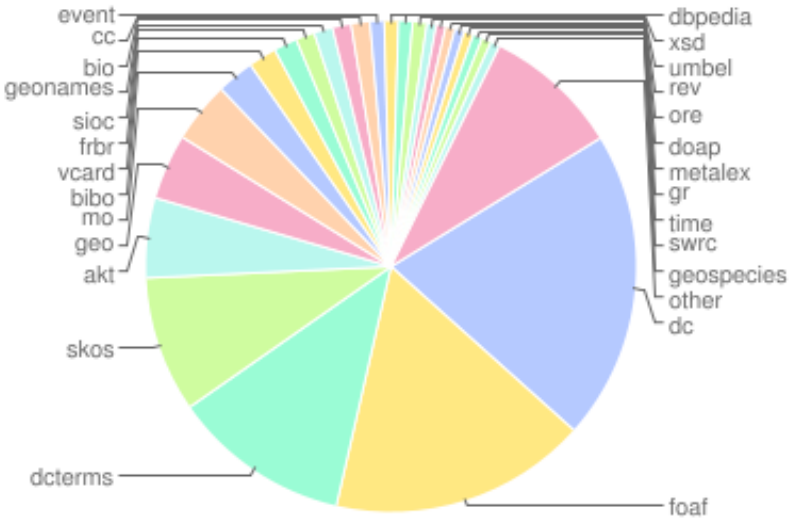


Abbildung 2.4: Die Verbreitung der bekanntesten Vokabelsammlungen in der Linked-Data-Cloud [8].

BIO

BIO beschäftigt sich ebenfalls mit der Beschreibung von Personen, geht dabei aber um einiges mehr ins Detail als *foaf* (siehe Abschnitt 2.5.1). Der Anwender kann unter anderem die Klassen und Eigenschaften *father*, *child*, *employer*, *divorce* und *retirement*[24] verwenden, um eine semantische Biographie einer Person zu erstellen.

Basic Geo

Das *Basic Geo*-Vokabular (geo) bietet *point*, *lat*, *long* und *alt*[25] für die standardisierte Ortsdefinition. Dies reicht aus, da über Längen- und Breitengrad jeder Punkt genau festgelegt werden kann.

[24] http://vocab.org/bio/0.1/.html
[25] http://www.w3.org/2003/01/geo/

Kapitel 3

Verwandte Arbeiten

Im folgenden Kapitel werden Arbeiten vorgestellt, die dem Themengebiet dieser Arbeit nahe stehen und neue interessante Ansätze liefern.

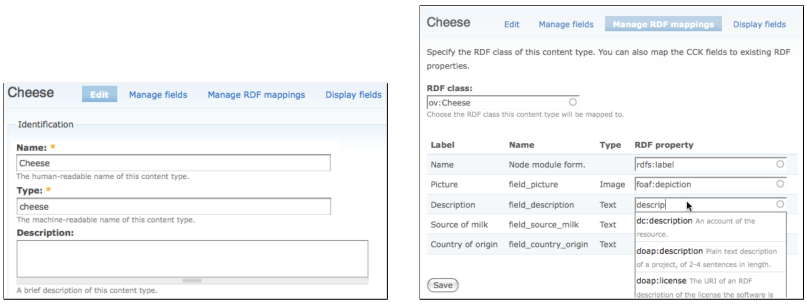
3.1 Drupal

3.1.1 Einführung

Schon im Jahr 2008 erklärte der Gründer von *Drupal*, Dries Buyteart, dass die Zeit des Semantic Web gekommen ist und die Community einen Weg finden sollte, das System in dieses neue semantische Zeitalter zu bewegen [10]. Daraufhin begann sich Stéphane Corlosquet mit einer Gruppe von Programmierern diesem Problem anzunehmen. Der erste große technische Report wurde im April 2009 veröffentlicht und widmete sich der Entwicklung und Integration eines semantischen Eingabeinterfaces als Modul für *Drupal* sowie ersten Usertests. Noch im selben Jahr wurden vier seperate Module fertiggestellt und auf der *8th International Semantic Web Conference* vorgestellt. Das *RDF CCK*-Modul ist das Herzstück der Erweiterung und ermöglicht die semantische Annotation von Texten. Über das *Evoc*-Modul kann der User externe Vokabeln in das eigene System laden. Der *RDF SPARQL Endpoint* bietet die eigenen semantischen Daten fremden Seiten an und über den *RDF SPARQL Proxy* kann man externes Wissen in die eigene Seite integrieren [13].

3.1.2 Content Types zu Semantic Objects

Einer der Gründe, warum die Integration in *Drupal* so schnell vorangegangen ist, ist sicherlich die Tatsache, dass das System von Grund auf sehr strukturiert aufgebaut wurde und damit einen guten Ausgangspunkt lieferte. Die User hatten bisher schon die Möglichkeit, neue Seitenelemente als sogenannte *Nodes* anzulegen und diesen einen speziellen *Content-Type* zuzuweisen, wofür wiederum eigene *Fields* definiert werden konnten.



(a) (b)

Abbildung 3.1: Illustriert den typischen Vorgang des Anlegens eines semantischen Inhaltselements [12]. (a) Der User legt den content-type fest. In einem weiteren Schritt werden die Felder angelegt und zugewiesen. (b) Diesen Feldern werden die RDF Eigenschaften zugewiesen.

Genau bei dieser speziellen Art der Inhaltselementverwaltung wurde nun angesetzt. Es galt, dem User die Möglichkeit zu geben, diesen Feldern und Knoten passende semantische Klassen und Eigenschaften zuzuweisen. Das entwickelte Modul kann sich daraufhin um die weitere Datenverwaltung kümmern, die Ausgabe mit RDFa anreichern und auch die URIs als ansprechbare URLs zur Verfügung stellen. Durch das zuvor erwähnte *Evoc*-Modul und die Möglichkeit, externe Beschreibungen importierten zu können, umgeht *Drupal* auch die Gefahr, ein in sich abgeschlossenes Vokabular zu erstellen, mit dem sonst niemand etwas anfangen kann.

3.1.3 Blockstein-Inhalt

Drupal ist einer der großen Vorreiter des Semantic Web – als eines der populärsten Content-Management-Systeme [13] bietet es erstmals einer großen Anzahl von normalen Webusern den Zugang zu semantischen Funktionalitäten.

Das eigens gesetzte Ziel war, den User nicht mit zu großem Arbeitsaufwand abzuschrecken. Eigene Studien haben ergeben, dass die ersten Durchgänge des Anlegens eines Inhaltselements mit semantischer Annotation im Vergleich zur alten Methode ohne semantische Informationen rund ein Drittel mehr Zeit in Anspruch nehmen [12]. Das sind gute Werte, die sich noch verbessern werden, sobald ein User den Vorgang verinnerlicht hat.

Es stellt sich aber unweigerlich die Frage, ob diese Art der Inhaltsverwaltung überhaupt sinnvoll ist. Der User hat hier nicht die Möglichkeit, einen Fließtext zu verfassen und sich danach um die Annotation zu kümmern. Es muss sofort bei der Planung des Inhalts bedacht werden, dass sich Texte aus einzelnen Blöcken zusammenbauen. Für Seiten mit vielen und längeren Artikeln kann dies durchaus eine große Barriere darstellen.

3.2 Loomp

3.2.1 Einführung

Bei *Loomp* handelt es sich um ein Websystem, das von Markus Luczak-Rösch und Ralf Hesse, Doktoranden der *Freie Universität Berlin*, entwickelt wurde. Hauptziel des Projektes war es, ein semantisches System zu entwickeln, bei der sich die Annotation von Objekten an den Formatierungsfunktionen von Word orientiert. Da nahezu jeder Webuser mit diesem Vorgang vertraut ist, wäre auch der Schritt zur semantischen Inhaltsanreicherung nicht mehr groß.

Loomp bildet dabei als eigenständiges System die Grundlage für verschiedene Module, die dem User die semantische Arbeit erleichtern sollen (siehe Abbildung 3.2). Elemente wie *Linked Data Client*, *Faceted Browsing* und *Faceted Viewing* sind so genannte Frontend-Anwendungen, die den Webanwender direkt ansprechen. Das Backend ist in die Module *One Click Annotator*, *Content Management* und *Vocabulary Management* sowie eine Serveradministration aufgeteilt, wobei speziell die zuletzt genannten Teilprogramme für die semantische Anreicherung interessant sind.

Bei der Entwicklung des Systems wurde besonders darauf geachtet, die Anwendung nicht für Semantic-Experten, sondern für normale Webredakteure zu konzipieren. Dabei lag der Fokus auf dem Design eines intuitiven Userinterfaces, bei dem der User nicht mit technischen Begriffen konfrontiert wird. Ein weiterer wichtiger Punkt war die Darstellung und Verwaltung des Vokabulars, bei dem der Redakteur auch wieder nur die nötigsten Informationen geliefert bekommt und zum Beispiel keine Namensräume angezeigt werden [16].

3.2.2 One Click Annotator

Das für diese Arbeit relevanteste Modul ist der sogenannte *One Click Annotator (OCA)*. Grundlage dafür bildet der TinyMCE WYSIWYG[1]-Editor [16]. Dabei wird die Erfahrung, die Computeruser mit *WYSIWYG*-Editoren wie Word und eben TinyMCE bereits gesammelt haben, herangezogen, um den User über das selbe Prinzip mit semantische Annotationen vertraut zu machen. Wenn der Redakteur also weiß, dass durch einen Klick auf den Button *italic* der Text kursiv geschrieben wird, soll er künftig auch über einen ähnlichen Vorgang semantische Informationen zuweisen können [24] (siehe Abbildung 3.3).

Der vom User erstellte Text wird schon während der Eingabe durch eine eigene Textanalyse, die jedoch auch von externen Diensten wie *OpenCalais*[2] unterstützt wird, untersucht, um dem User in der Leiste über dem Textfeld passende Annotationen vorzuschlagen. Werden jedoch die gewünschten se-

[1] WYSIWYG ist das Akronym für *What You See Is What You Get*
[2] Weitere Informationen: http://www.opencalais.com/

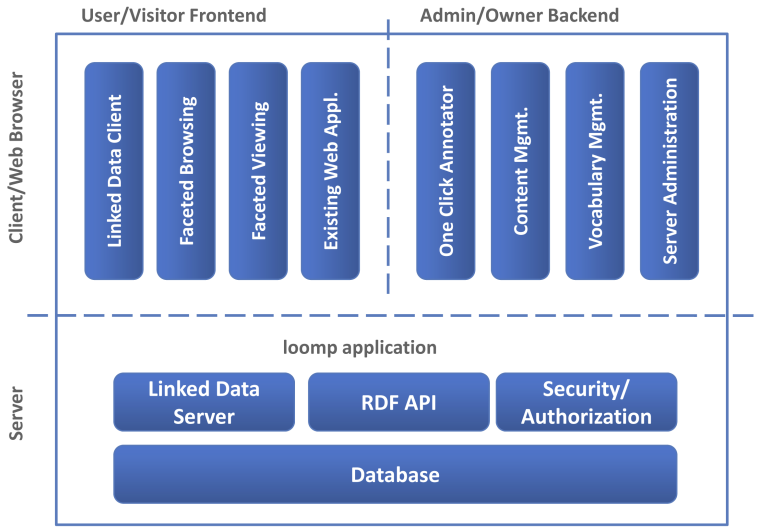


Abbildung 3.2: Loomp Systemarchitektur [16].

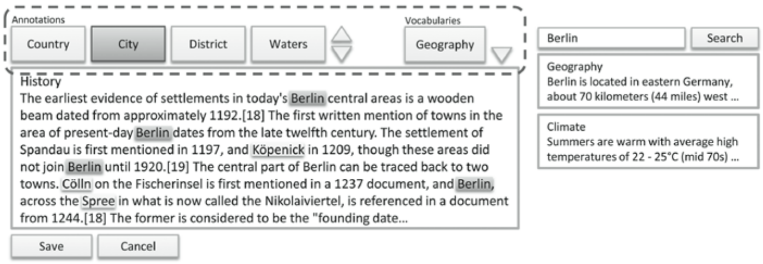


Abbildung 3.3: One Click Annotator [24].

mantischen Elemente nicht gefunden, hat der User zusätzlich die Möglichkeit, die Datenbank nach den passenden Objekten zu durchsuchen. Im nächsten Schritt wird eine Liste mit passenden Elementen samt Attributen zur Wahl gestellt. Spezifische Objektvorschläge werden im Editor durch spezielle Hervorhebungen der Texte angezeigt [24].

3.2.3 Der perfekte Ansatz?

Loomp bietet dem User mit dem *One Click Annotator* eine benutzerfreundliche Oberfläche, die speziell darauf ausgelegt ist, dass User ohne Vorwissen einen Einstieg in die semantische Welt finden. Das System selbst steht auf der offiziellen Webseite[3] zur Zeit[4] nicht zum Download zur Verfügung. Die

[3]Weitere Informationen: http://www.loomp.org/
[4]Stand: 25.05.2011

Demo wurde nach einer Testphase wieder offline genommen und es wird an einer neuen Version gearbeitet.

Das Projekt bietet sehr interessante Ansätze, speziell mit der Fokussierung auf normale Webanwender, die kaum Grundwissen in diesem Bereich mitbringen. In den meisten Semantic Web-Applikationen, die zur Zeit im Umlauf sind, ist grundsätzlich ein erhöhter Know-How-Level nötig, um diese überhaupt zu verstehen [14]. Die Entwickler haben sich diesem Problem gut angenommen und versucht, die Funktionen so simpel wie möglich zu gestalten. Problematisch könnte das als Grundlage der Applikation entwickelte Content-Management-System sein. Obwohl es zum aktuellen Zeitpunkt nicht mehr zum Download steht, ist davon auszugehen, dass es dem Vergleich bezüglich Funktionalitätsumfang mit gängigen Systemen wie *Drupal*, *Wordpress* oder *Typo3* kaum standhalten wird. Nicht ohne Grund sagen die Entwickler selbst, dass die bisherige Demo als *proof-of-concept* ausreichte, der Ansatz aber als Produktivanwendung noch nicht funktioniert und nun an einer Version gearbeitet wird, die eigenständig als API mit externen Editoren zusammenarbeitet[5].

3.3 Onthologische Wissenssammlung

Dieser Abschnitt beschäftigt sich mit dem alternativen Ansatz einer Wissensdatenbank als Grundlage einer Webseite und stellt verschiedene Beispiele vor.

3.3.1 Allgemein

Um semantisches Wissen ins Netz zu bekommen, kann man die Vorgehensweise der Dateneingabe und Verarbeitung auch komplett umdrehen. Der gängige Ablauf bei Design und Aufbau von Webseiten ist normalerweise die Umsetzung des Designs mit variablen Technologien und Systemen sowie die darauf folgende Befüllung der Webseite. Dafür wird der gewünschte Text auf den zugeordneten Seiten platziert. Dieser Vorgang ist, wenn man es auf die nötigsten Prinzipien zusammenfasst, ziemlich seitenstruktur- und designorientiert.

Als alternativer Ansatz bietet sich jedoch an, zuerst die nötigen Daten in einer Onthologiedatenbank aufzubauen und die Struktur und Ausgabe der Webseite von den vorhandenen Daten abhängig zu machen.

OntoWebber

Der sogenannte *OntoWebber* wurde an der *Stanford University* entwickelt und hat genau diesen Ansatz als Grundlage des Systems. Im Fokus der Ar-

[5] http://tinyurl.com/42t34ln, Kopie auf CD-ROM (docs/loompBlog.pdf)

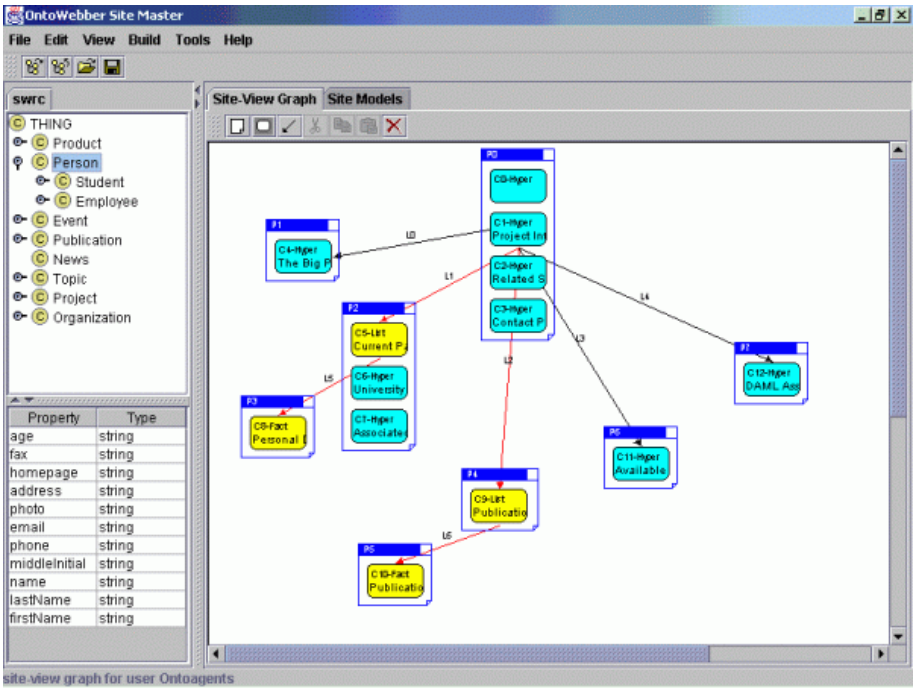


Abbildung 3.4: OntoWebber: Über den site-view-graph wird festgelegt, welche Informationen und Elemente auf der Seite wie ausgegeben werden sollen. Die hellblauen Elemente repräsentieren dabei sogenannte *Cards*, die spezifische Daten enthalten. Diese werden in *Pages* (dunkelblauer Rahmen) gruppiert und in dieser Struktur dann auch auf der Seite ausgegeben. Mit *Links* kann auf weitere Seiten und Objekte verlinkt werden.

beit lag dabei, ein System zu entwickeln, mit dem es möglich ist, große Datenmengen zu verwalten. Die Administratoren sollen dabei so viel Spielraum wie möglich bei der Darstellung und Personalisierung der Webseite erhalten [22].

Nachdem die Onthologie ins System importiert wurde, setzt dieses stark auf sogenannte Ansichtsmodelle. Unter anderem hat der Redakteur die Möglichkeit, im *navigation model* festzulegen, über welche Elemente der User auf der Seite navigieren kann. Das *content model* legt hingegen fest, wie die semantischen Informationen auf der Seite strukturiert und ausgegeben werden sollen. Der Vorgang wird in Abbildung 3.4 genauer beschrieben[6].

Der *OntoWebber* bietet eine Vielzahl an Ansätzen und Möglichkeiten, ist allerdings als etwas älterer Prototyp zu verstehen. Für normale Enduser ist er nicht geeignet, da diese weder das nötige Vorwissen in onthologischer Aufbereitung der Informationen noch das Verständnis der komplexen Model-View Methodik mitbringen werden.

[6] http://infolab.stanford.edu/OntoAgents/OntoWebber/index.html

Abbildung 3.5: Rechts befindet sich die Inhaltsverwaltung und links die dazu passende Vorschau von *Webnodes*.

Webnodes

Ein kommerzielles System, das mit ähnlichen Ansätzen und Ideen arbeitet, ist *Webnodes*. Dieses versteht sich als ASP.NET Content-Management-System und hat sich selbst zum Grundsatz gemacht, dem User die Möglichkeit zu geben, seinen Inhalt zu verwalten – und nimmt damit die Fokussierung von der herkömmlichen Seitenstrukturverwaltung weg.

Der Redakteur kann durch die Funktionalitäten des Systems eine Wissensdatenbank mit Klassen, Objekten und Eigenschaften anlegen und diese miteinander in Verbindung setzen (siehe Abbildung 3.5). Durch diese Verknüpfungen entsteht eine natürliche Seitenstruktur, die daraufhin als Grundlage für den Aufbau der Webseite genommen wird.

Weitere Informationen zu diesem kostenpflichtigen CMS gibt es auf der offziellen Homepage[7].

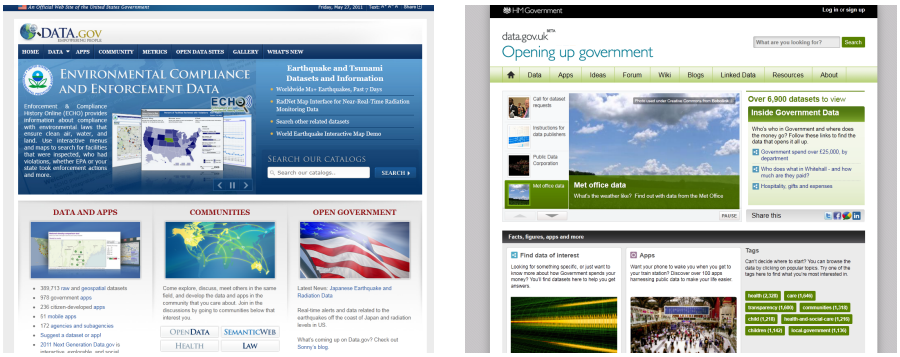
3.3.2 E-Government

Auch Regierungen weltweit beginnen, die Vorteile des Semantic Web für sich zu entdecken (siehe Abbildung 3.6).

England hat mit data.gov.uk eine Initiative gestartet, um öffentliche Daten für die Bevölkerung leichter zugänglich zu machen. Dabei wird von dem Grundgedanken ausgegangen, dass alle Informationen, die unter einem *Freedome of Information*-Antrag zwangsläufig offengelegt werden müssen, ohne jegliche Formalitäten jederzeit erreichbar sein sollten [11].

Auch die US-Regierung unter Präsident Obama hat sich vorgenommen, die Informationspolitik zu lockern und www.Data.gov ins Leben gerufen [6].

[7] http://www.webnodes.com/



(a) (b)

Abbildung 3.6: E-Government Portale. (a) US (b) UK

Obwohl beide Projekte noch in Kinderschuhen stecken, ist die Intention, Daten für Bürger schneller, einfacher und unbürokratischer zur Verfügung zu stellen, lobenswert – auch unter der Berücksichtigung, dass dadurch die Informationsdichte der Linked-Data-Cloud wächst. Skeptischen Betrachtern kann dabei die Frage aufkommen, warum man erst mit dem Fortschritt des semantischen Web anfängt, diese Daten zugänglicher zu machen.

Hier ist eine Antwort allerdings schnell gefunden. Durch den strukturellen Aufbau semantischer Informationsdatenbanken ist es leichter, dieses Wissen auch verständlich und strukturiert auf einer Seite auszugeben. Vor allem bietet es aber die Möglichkeit, die interne als auch externe Datenverwaltung in einem Arbeitsschritt zu erledigen. Sobald bestimmte Informationen digitalisiert sind, wandern sie in die Datenbank und damit, je nach Konfigurationen, sofort zur Linked-Data-Cloud.

Damit profitieren Regierungen nicht nur von einer offenen Informationspolitik [6], sondern auch von gut strukturierten Datenbanken, die für die eigenen Recherchen enorme Zugewinne bringen kann.

3.3.3 BioPortal

Einen weiteren Ansatz verfolgen Wissenssammlungen wie das medizinische Webportal *BioPortal*. Speziell im Bereich Medizin gibt es mittlerweile eine unüberschaubare Menge an Onthologien, die von verschiedensten Organisationen ins Netz gestellt werden [26]. Hintergrund dieser Onthologien ist der Wille, vorhandenes Wissen möglichst erreichbar und öffentlich zugänglich zu machen und damit die Forschung bestmöglich zu unterstützen. Nun sind diese Onthologien für sich gesehen sicherlich schon eine große Hilfe; eine ganz andere Dimension wäre es aber, wenn dieses Wissen an einem Ort gesammelt aufzufinden ist. Genau hier setzt *BiopPortal* an und bietet als Repository über 200 biomedizinischen Onthologien und Terminologien [26].

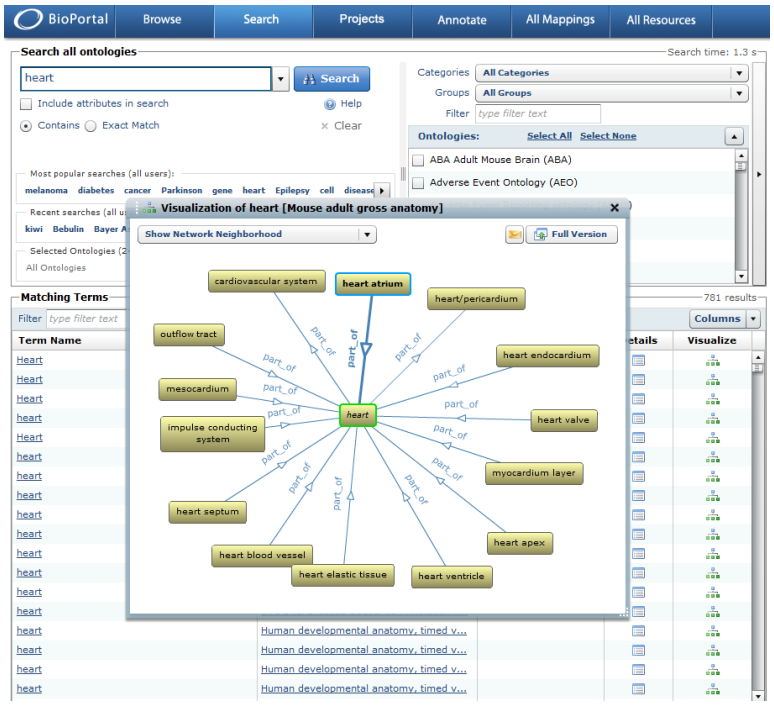


Abbildung 3.7: *BioPortal*-Suchinterface mit Visualiserung des Terms *Heart* aus der *mouse adult gross anatomy*.

Um das System so kompatibel und integrationsfreundlich wie möglich zu gestalten, unterstützt es mehrere *Knowledge Representation Languages*, wie OBO Format, Protégé Frames, RDF und OWL [28]. Der Inhalt der Ontologien als auch die dazugehörigen Metadaten, wie *Keywords*, *Beschreibungen*, *Versionsinformation*, *Veröffentlichungsdatum* und *Onthology Autor* können automatisch oder durch Usereinsendung aktualisiert werden [27]. User haben im Weiteren ebenfalls die Möglichkeit, verschiedene Onthologien durch *Links* miteinander zu verbinden und so den Informationsgehalt zu erhöhen. Um die eingetragenen Daten zu durchsuchen, bietet *BioPortal* eine ausgeprägte Suche mit verschiedensten Funktionalitäten (siehe Abbildung 3.7).

Einen innovativen Ansatz bietet das System mit der Einbindung der Community und verbindet damit das semantische Web mit Web 2.0 Funktionalitäten. User können bei allen Onthologie-Klassen im System sogenannte *Notes* hinterlassen [27]. Darin können Kritik, Anmerkungen oder Verbesserungsvorschläge hinterlassen werden, auf die andere User ihrerseits wieder eingehen können. So ist es möglich, eine langgezogene Korrespondenz über Email zu umgehen und über diese Form der direkten Kommunikation zwischen mehreren Usern und Onthologie-Autoren etwaige Fehler schnell aufzudecken und zu beheben.

```
{
   "name": "Facebook Platform",
   "type": "page",
   "website": "http://developers.facebook.com",
   "username": "platform",
   "founded": "May 2007",
   "company_overview": "Facebook Platform enables anyone to build...",
   "mission": "To make the web more open and social.",
   "products": "Facebook Application Programming Interface (API)...",
   "fan_count": 449921,
   "id": 19292868552,
   "category": "Technology"
}
```

Programm 3.1: Open Graph - Knoten von *Facebook*, erreichbar unter https://graph.facebook.com/19292868552.

3.4 Weitere Ansätze und Projekte

3.4.1 Facebook - Open Graph

Im April des Jahres 2010 kündigte Mark Zuckerberg, CEO von Facebook[8], auf der *f8 developer conference* an, dass das *Social-Networking*-Portal künftig auf das RDF-basierende *Open Graph*-Prokoll setzen wird [10]. Dabei nannte er diese Entwicklung „the most transformative thing we've ever done for the Web[9]."

Auch wenn es sich bei dieser Äußerung in bestimmten Maße natürlich um eine Marketingformulierung handelt, ist die Aussage dahinter sicherlich nicht ganz falsch. Die Plattform ändert im Zuge dieser Umstellung die Art und Weise, wie Daten gespeichert werden – jeder Datensatz bekommt eine eindeutige ID, über die er künftig im Open-Graph auffindbar sein wird (siehe Abbildung 3.1). Damit ist es aber noch nicht getan. Ein neues Feature, das *Facebook* im Rahmen dieser Umstellung vorgestellt hat, ist die Möglichkeit, den bekannten *Like*-Button auch auf eigene Webseiten und Inhalte zu stellen – durch diese Einbindung werden diese Ressourcen Teil des Open Graph. Wenn nun, zum Beispiel, ein User auf der Filmdatenbank imdb[10] bei dem Film *König der Löwen* diesen *Like*-Button drückt, wird der Vorgang nicht nur auf dessen *Facebook-Wall* angezeigt, sondern direkt unter *„Filme, die mir gefallen"* eingetragen. Das System handhabt daher den Film als eindeutiges Objekt – und genau das ist auch die Grundidee des Semantic Web.

Hinter diesem Umbau steckt wahrscheinlich nicht die Intention von *Fa-*

[8] www.facebook.com

[9] http://news.cnet.com/8301-13577_3-20003053-36.html - Kopie auf CD (docs/FacebookF8.pdf)

[10] www.imdb.com

cebook, die Linked-Data Bewegung voranzutreiben, sondern lediglich eine bessere Möglichkeit, die internen Daten zu archivieren [5]. Nichtsdestotrotz macht das Portal damit einen großen Schritt für das Semantic Web. Millionen von Webseiten werden anfangen, Inhalte über die Plugins semantisch aufzubereiten. Alleine in der ersten Woche wurden schon 50.000 Implementierungen gemeldet[11]. Infolgedessen werden eine enorme Menge an semantischen Daten die Linked-Data-Cloud berreichern – auch wenn diese vorerst noch durch das beschränkte Vokabular von *Facebook* eingegrenzt wird.

Anzumerken sind in diesem Fall natürlich trotzdem die Bedenken des Datenschutzes, denn auch mit dieser Entwicklung werden die Kritiker immer lauter[12][13].

3.4.2 Semantic Pingback

Bei *Semantic Pingback* handelt es sich um ein Projekt, welches sich zum Ziel gesetzt hat, die Verbreitung des semantischen Web durch einen Dienst zu beschleunigen, der auch die Blogosphere entscheidend verändert hat [32]: Pingback[14]. Grundgedanke dahinter ist die Tatsache, dass der Enduser im Semantic Web bisher kaum eine Motivation dafür hatte, seine Daten für die Linked-Data-Cloud freizugeben, außer den Wunsch, dass seine Daten darin verfügbar sind und zu einem gewissen Grad auch Idealismus, um bei dieser neuen Entwicklung dabei zu sein. Sobald die Daten einmal veröffentlicht wurden, muss der User damit leben, dass er keine weiteren Informationen über den Verbleib der Daten bekommt. Es gibt keine weiteren Informationen, wo die Daten benutzt werden und wie oft die Daten von Quellen bezogen werden[33].

Semantic Pingback setzt nun bei diesem Problem an und führt neue Funktionen ein, um die Userexperience des semantischen Webs und damit die Teilnahmebereitschaft als auch den Nutzen für Redakteure zu steigern. Sobald der Service von Datenersteller und Datennutzer verwendet wird, kann nun nachvollzogen werden, wie sich die Daten im Internet verbreiten und durch Links wird offen auf die jeweils andere Stelle verlinkt (siehe Abbildung 3.8). Dadurch kann der Datenersteller nicht nur nachvollziehen, wofür seine Daten genutzt werden, sondern profitiert auch von neuen Besuchern.

Das System bietet neben vielen weiteren Funktionen, die auf der offiziellen Homepage genauer nachzulesen sind[15], auch Möglichkeiten, Spam und Datenmissbrauch zu vermeiden.

Bleibt abzuwarten, ob ein Service dieser Art großflächig bei semantischen Systemen integriert wird.

[11] http://tinyurl.com/wallblog, Kopie auf CD-ROM (docs/TheWallBlog.pdf)
[12] http://tinyurl.com/SB-facebookFailure, Kopie auf CD-ROM (docs/FBFailure.pdf)
[13] http://tinyurl.com/SB-FacebookDevil, Kopie auf CD-ROM (docs/FBDevil.pdf)
[14] http://www.hixie.ch/specs/pingback/pingback-1.0
[15] http://aksw.org/Projects/SemanticPingBack

Abbildung 3.8: Beispiel für eine Oberfläche mit *Semantic Pingback* [33].

3.4.3 Triplify

Einen weiteren Ansatz bietet das Projekt *Triplify*. Es handelt sich hierbei um ein sogenanntes *Database-Mapping-Tool*, das es ermöglicht, existierende Datenbanken und deren Inhalt zu verwenden, um daraus semantische Inhalte zu generieren. Das Programm ist dabei so ausgelegt, dass es leicht in bestehende Systeme integriert werden kann. Der Grundgedanke dahinter ist, dass weltweit die Mehrheit der für das Internet relevanten Daten in relationalen Datenbanksystem wie *MySQL* abgelegt sind. Anstatt von den Redakteuren zu verlangen, auf eine komplett neue Inhaltsaufbereitung umzusteigen und die alten Daten alle noch einmal überarbeiten zu müssen, arbeitet dieses System mit der existierenden Datenstruktur [3].

Voraussetzung dafür ist selbstverständlich, dass die bisher verwendete Datenstruktur so gut aufgebaut ist, dass man daraus semantische Objekte und Klassen ablesen kann. Man weist dabei den jeweiligen Tabellen und Feldern die benötigten semantischen Annotationen zu, woraufhin das System die Onthologie automatisch erstellt. Dabei wird aber schon die Grenze dieses Ansatzes klar – Fließtexte, die als solche auch in der Datenbank gespeichert wurden, können damit nicht tiefgehend annotiert werden.

3.4.4 Scotty

Scotty steht für *Semantic Category, Ontology, Tag and Taxonomy sYstem* und ist ein laufendes Projekt[16], das es sich zum Ziel gesetzt hat, semantische Funktionen in das Web-Content-Management Typo3 zu integrieren. Zur Zeit sind noch keine relevanten Ergebnisse bekannt.

[16] Offizielle Scotty-Homepage: http://forge.typo3.org/projects/extension-scotty

Kapitel 4

Anforderungsanalyse

4.1 Typo3 – Content Management System

4.1.1 Beschreibung

Bei Typo3 handelt es sich um ein Content Management System, das ursprünglich von Kasper Skårhøj entwickelt wurde. Mit über 500.000 Exemplaren im Einsatz[1] gehört es zu den weltweit am meisten verbreiteten Systemen. Es handelt sich um ein Freeware-Produkt, was bedeutet, dass für die Anwender keinerlei Lizenzkosten anfallen. Weiterhin basiert Typo3 auf der OpenSource-Philosophie – der Quellcode ist somit für jeden zugänglich und erlaubt es der Community, Typo3 den individuellen Wünschen anzupassen und weiterzuentwickeln.

Das Content-Management-System ist für seine Größe und Vielfältigkeit bekannt und daher auch eher bei großen Webprojekten und Firmenauftritten und nicht bei kleinen Webblogs zu finden. Dafür gibt es passendere Systeme mit weniger Overhead, wie zum Beispiel *Wordpress* oder *Drupal*.

Die grundsätzliche Systemstruktur von Typo3 baut sowohl auf Frontend- als auch auf Backend-Funktionalitäten, die an sich strikt voneinander getrennt sind. Das sogegannte *Frontend* umfasst den gesamten optischen Output der Webseite. Unabhängig von Typo3 kann der Webdesigner ein HTML Template erstellen, das das Seitenlayout vorgibt. Darin werden dann mit sogenannten *Ankern*, zum Beispiel *'###content###'*, die Inhalte aus dem Typo3 in das Seitenlayout eingebunden.

Im sogenannten *Backend* verwaltet der Redakteur den kompletten Seiteninhalt. Hier wird die Seitenstruktur definiert, der Aufbau der Menüs und die konkreten Inhaltsseiten sowie auch Seiteninhalte wie Texte, Bilder und weitere Inhalte wie PlugIns verwaltet. Wie die Texteingabe in Typo3 genau abläuft, ist in Abschnitt 4.1.2 zu finden.

[1] http://www.slideshare.net/benvantende/typo3-presentation-cebit-2010, Typo3 Präsentation - CeBit2010, Kopie auf CD-ROM (Datei docs/TYPO3presentationCeBIT2010.pdf vom 15.05.2011).

Die Einarbeitungszeit für Entwickler kann bei Typo3 durch den komplexen Aufbau mehrere Wochen betragen; Anwender beziehungsweise Redakteure kommen meist nach einer kurzen Einführungsphase von wenigen Stunden mit den benötigten Funktionen zurecht. Typo3 wurde in der Vergangenheit trotzdem wegen der komplexen Bedienung kritisiert - ein Vorwurf, der durch vereinfachte Menüführungen und Funktionen wie Front-End-Editing [2] in neueren Versionen ausgeräumt werden sollte.

Ein wichtiger Bestandteil von Typo3 sind auch die sogenannten Extensions (*Erweiterungen*), die es Webentwicklern erlauben, schnell und einfach eigene Module für Typo3 zu entwickeln und damit das System exakt auf die Bedürfnisse der Anwender anzupassen. Genau hier wird auch angesetzt, um eine semantische Erweiterung für Typo3 zu entwickeln, die bisher[3] in dieser Form noch nicht existiert.

Typo3 basiert auf der Skriptsprache PHP und die Daten werden in einer relationalen Datenbank gespeichert. Hierfür wird meist MySQL eingesetzt, es kann aber auch mit Alternativen wie PostgreSQL oder Oracle gearbeitet werden.

4.1.2 Texteingabe in Typo3

Um eine semantische Erweiterung für Typo3 entwickeln zu können, ist es vor Beginn erst einmal wichtig zu verstehen, wie die normale Texteingabe in Typo3 überhaupt funktioniert und wie das System mit den Daten anschließend umgeht. Zur Veranschaulichung des Vorgangs zeigt Abbildung 4.1 eine Illustration, die den genauen Arbeitsablauf bei einer Texteingabe Schritt für Schritt erläutert.

Die Daten eines Textblock-Elements werden im Hintergrund in der Datenbank in der Tabelle *tt_content* in der Spalte *bodytext* gesammelt abgespeichert.

4.2 Anforderungen

Ziel ist es, semantische Funktionalität in das bestehende Typo3 System zu integrieren. Bei der Entwicklung wird mit Version 4.2.8 von Typo3 gearbeitet. Die Erweiterung sollte aber auch möglichst in späteren Versionen ohne größere Probleme laufen, wenn die grundsätzliche Systemstruktur nicht verändert wird. Primäres Ziel ist das Erstellen einer Wissensbasis. Der User soll daher die Möglichkeit haben, den Inhalt mit semantischen Informationen anzureichern.

[2] Front-End-Editing erlaubt dem User eine vereinfachte Administrierung der Inhalte direkt über das Front-End der Webseite. Dabei werden dem Redakteur, sobald er in Typo3 eingelogged ist, bei den jeweiligen Inhaltselementen die Möglichkeit gegeben, diese direkt mit einem Klick zu bearbeiten.

[3] Anfang 2011

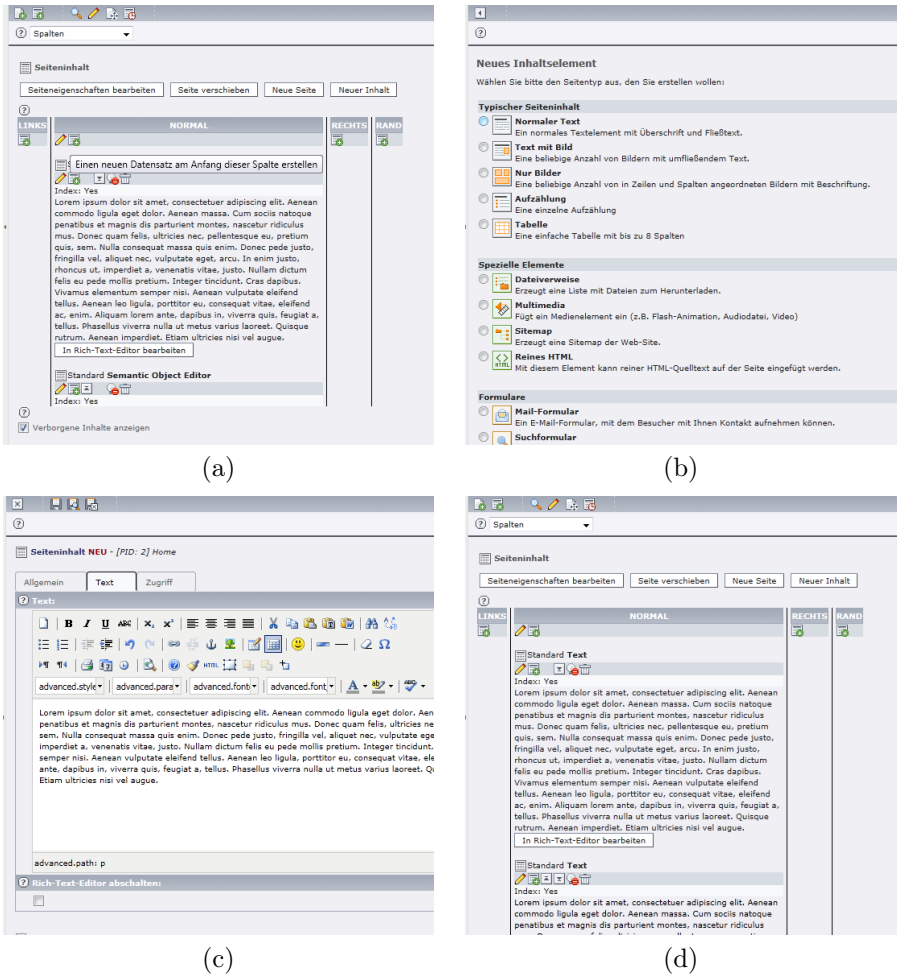


Abbildung 4.1: Illustriert den typischen Vorgang des Anlegens eines Textelements. (a) Der User wählt in der Spalte, in der der Inhalt später auf der Webseite angezeigt werden soll, die Option *neuen Datensatz*. (d) Im nächsten Schritt wird der Typ des Inhaltselementes ausgewählt - in diesem Fall *normaler Text*. (c) Nun folgt die eigentliche Eingabe des Inhaltes durch den User im Rich-Text-Editor Fenster. (d) Sobald der User den Vorgang abgeschlossen und das neue Inhaltselement abgespeichert hat, wird es nun in der Spalte angezeigt.

4.3 Anwendungsfälle

Um eine semantische Erweiterung für Typo3 entwickeln zu können, war es erst einmal wichtig, die Grenzen des Projekts abzustecken und festzulegen, welche Funktionalitäten angeboten werden müssen.

4.3.1 Texteingabe

Die Grundlage der Erweiterung ist die simple Funktionalität der Texteingabe. Dabei stellt sich unweigerlich die Frage, ob man sich an der inhaltlich modularen Struktur von Drupal orientiert (siehe Abschnitt 3.1).

Der Vorteil dabei wäre eine übersichtlichere Strukturierung der Inhaltselemente. Dies würde allerdings eine komplette Überarbeitung des Typo3-Kernsystems nach sich ziehen, womit auch die Gefahr bestünde, dass man die Erweiterung in neuere Typo3-Versionen mit ziemlicher Sicherheit nicht oder nur schwer übernehmen kann.

Ein weiterer wichtiger Punkt, den man bei diesen Überlegungen nicht außer Acht lassen kann, sind die Anwender und Redakteure, die später mit dem System arbeiten sollen. In den meisten Fällen sind diese mit dem Arbeitsablauf in Typo3 vertraut und eingearbeitet. Für diese User wäre es eine enorme Umstellung, wenn man für ein semantisches Modul die komplette Logik hinter der inhaltlichen Aufbereitung der Daten umbauen würde. Erschwerend kommt noch hinzu, dass die restlichen Funktionalitäten von Typo3 in diesem Fall nicht behindert oder gar zerstört werden dürfen.

Damit lässt sich zusammenfassend sagen, dass die modulare Aufbereitung für Drupal natürlich funktioniert, man diesen Ansatz aber nur schwer auf Typo3 umlegen kann. Die grundlegende Logik hinter diesen Systemen ist komplett unterschiedlich und daher ist es auch nötig, hier einen anderen Ansatz zu entwickeln, um zu einem zufriedenstellenden Ergebnis für Entwickler und Redakteure zu kommen.

Das bedeutet, dass man sich an der aktuellen Texteingabe orientieren sollte, um gewährleisten zu können, dass das System weiterhin stabil läuft, die Erweiterung durch ein System-Update nicht sofort unbrauchbar wird und man zusätzlich garantieren kann, dass eingarbeitete User mit dem Arbeitsablauf der Erweiterung nicht vollkommen überfordert werden.

4.3.2 Semantisches Anreichern von Inhalten

Bei der Überlegung, wie man Inhalte mit semantischen Informationen am besten anreichern kann, ist es zunächst nötig, die bisherigen Informationen zu berücksichtigen. In Kapitel 3.3 wurde der Ansatz der onthologischen Content-Management-Systeme vorgestellt. Da dieser Ansatz aber mit Typo3 nicht vereinbar ist, fällt er aus den Überlegungen heraus.

Wie in Abschnitt 4.3.1 bereits besprochen, sollte die in Typo3 übliche Struktur der Texteingabe möglichst nicht geändert werden. Damit stellt sich die Frage, wie man die semantische Anreicherung angehen sollte, was der User für Anforderungen an dieses System stellen kann und was es für Grenzen bietet.

Zielsetzung der Arbeit ist, Inhaltstexte mit semantischen Informationen auszustatten und nicht nur eine reine Wissensdatenbank zu erstellen. Auf-

wendige, eigenständige Module wie Protégé[4] bieten dem User zwar umfangreiche Möglichkeiten, Wissen über semantische Funktionen auszudrücken, lassen sich aber weder mit der Inhaltseingabe noch mit der Struktur von Typo3 vereinen. Würde man ein ähnliches Modul für Typo3 umsetzen, müsste der User seine Inhalte doppelt eingeben und verwalten, da eine direkte Fließtextein- als auch -ausgabe über diesen Ansatz nicht möglich ist. Im Weiteren lässt sich auch mit Sicherheit sagen, dass ein Modul mit derartig vielen Funktionen den User überfordern, verwirren und abschrecken könnte. Ein Modul dieser Art würde daher den Rahmen des erstrebten Projektziels sprengen, als auch die Grundintention, einen leicht verständlichen Einstieg in die semantische Welt zu bieten, verfehlen.

Es muss daher ein simpler Weg gefunden werden, wie die semantische Informationsanreicherung der Redaktionstexte direkt in die vom User gewohnte Texteingabe-Umgebung integriert werden kann, ohne diesen mit neuen Funktionen sofort wieder abzuschrecken. Hier ist es wichtig, eine Balance zwischen Funktionalität und Userexperience zu finden und die Funktionen des Programmes auf eine überschaubare Anzahl einzuschränken.

Zudem dürfen Inhalte auf keinen Fall durch die semantische Anreicherung von künftigen Anpassungen gesperrt werden. Gerade auf Webseiten ist es wichtig, dass Texte immer aktuell sind und es muss unbedingt gewährleistet werden, dass der Redakteur die zuvor verfassten Texte auch weiterhin bearbeiten kann, ohne die ganze Semantik der Inhalte zu zerstören.

Um den Anwender nicht zu verwirren, ist auch sicherzustellen, dass die semantischen Informationen übersichtlich aufgearbeitet werden. Der User muss die Möglichkeit haben, die erstellte Semantik jederzeit selbst zu verstehen, im Nachhinein noch bearbeiten und wieder löschen zu können.

4.3.3 Wissensdatenbank

Bei der Anforderungsanalyse wurde festgelegt, dass die Fließtext-Eingabe von Typo3 nicht verändert wird. Da man davon ausgehen kann, dass die Redakteure in ihrem Fließtext immer mehr als nur ein semantisches Element einbauen, muss auch bei der Konzeption der Wissensdatenbank darauf Rücksicht genommen werden, dass Fließtext und Wissen nicht in einer Tabelle zusammen abgespeichert werden können.

Internes Wissen

Es ist darauf zu achten, dass die zu entwickelnde Erweiterung das semantische Wissen in eigenen Tabellen verwaltet. Die genaue Datenstruktur bleibt grundsätzlich offen (und wird bei der Umsetzung näher behandelt, siehe Kapitel 6), es ist jedoch darauf zu achten, dass man sich an die semantischen

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

Grundlagen hält und die Daten in einer Triple-Form abspeichert, um später damit auch eine sinnvolle Ausgabe erarbeiten zu können.

Externes Wissen

Zusätzlich wäre es von Vorteil, wenn der User die Möglichkeit hat, externe Wissensquellen zu nutzen, um vordefinierte Klassen in das System zu laden. Hier ist es wichtig, klarzustellen, dass es sich bei diesem Punkt nicht um externes Wissen in Form von Wissensdaten handelt, sondern rein um vordefinierte Klassen und Objekte wie foaf[5].

4.3.4 URI Bereitstellung

Obwohl es nicht zwingend nötig ist, dass die erstellten URIs wirklich im Web erreichbar sind, ist es doch „good practice“ und gern gesehen, wenn diese Option zur Verfügung gestellt wird [29, S. 66]. Da die vom Redakteur erstellte Semantic sowieso in der Wissensdatenbank abgespeichert wird, sollte es daher kein großes Problem darstellen, dieses Wissen auch mit einer im Web erreichbaren URI auszustatten.

Im System selber ist darauf zu achten, dass die Namensräume und URIs eindeutig bleiben - bei externen Quellen kann dies natürlich nicht garantiert werden.

4.3.5 Ausgabe

Damit das vom User angelegte Wissen auch seinen Weg ins World Wide Web findet, muss eine optimale Möglichkeit gefunden werden, dieses auszugeben. Hier sollten zwei Ausgabeformen berücksichtigt werden: die direkte Ausgabe in HTML und die Ausgabe der Informationen, wenn man eine URI als URL anspricht.

HTML Ausgabe

Bei der HTML Ausgabe ist besonders darauf zu achten, dass der normale User von den Ausgaben der semantischen Informationen nichts mitbekommt. Die Inhaltsseiten sollen genau so aussehen, wie der Redakteur diese zuvor bei der Inhaltsbefüllung in der Texteingabe definiert hat.

Wie bei den Grundlagen (siehe Kapitel 2) schon besprochen wurde, ist die einfachste Möglichkeit, semantische Informationen direkt in die HTML Ausgabe der Seite zu bekommen, der Webstandard RDFa. Damit kann man sichergehen, dass die gewünschten Informationen direkt auf der Inhaltsseite zu finden sind, auf der sie vom User definiert wurden und kein zusätzliches RDF Dokument geöffnet werden muss.

[5] Für weitere Informationen siehe http://www.foaf-project.org/.

Abbildung 4.2: Sceenshot DBpedia.

Named Graph Ausgabe

Eine weitere Ausgabe betrifft den Named Graph beziehungsweise die angresprochene URI. Obwohl es nicht zwingend benötigt wird, sollte die Erweiterung es ermöglichen, die erarbeiteten Informationen auch direkt über die URI aufzurufen. Als Vorlage für die Ausgabe wird sich an DBpedia[6] orientiert (siehe Abbildung 4.2).

Auch hier sollten die semantischen Informationen maschinenlesbar sein und daher ebenfalls per RDFa ausgegeben werden.

[6] http://dbpedia.org/page/Linz

Kapitel 5

Design

Dieses Kapitel befasst sich mit verschiedenen Entwürfen für das semantische Userinterface. Nachdem die Entwürfe vorgestellt wurden, werden sie in einer Diskussion miteinander verglichen, um an die beste Lösung für das Projekt zu kommen. Es wurde beim grafischen Design zwar Rücksicht auf die Anforderung genommen, die Texteingabemodalität von Typo3 zu belassen, die technische Umsetzbarkeit wurde aber noch nicht einbezogen, um die Ideen nicht sofort im Keim zu ersticken. Etwaige Problematiken der Entwürfe werden in der Diskussion angesprochen.

5.1 Interface Entwürfe

Es werden drei potentielle Entwürfe für ein semantisches Userinterface in der gewohnten Typo3-Texteingabe-Umgebung vorgestellt.

5.1.1 Semantic RTE

Bei dem Entwurf mit dem Titel *Semantic RTE* (siehe Abbildung 5.1) handelt es sich um den frühesten Entwurf des Userinterfaces. Orientiert wurde sich dabei am aktuellen Design des Rich-Text-Editors *tinyRTE*. Der Grundgedanke hinter dem Entwurf ist, die Funktionalität eines normalen RTE mit semantischen Funktionen zu erweitern. Das würde schlussendlich bedeuten, dass das RTE-System und das semantische System zu einer einzelnen Erweiterung verschmelzen.

Object-Handling

Um Objekte anzulegen, muss man den gewünschten Textteil markieren und daraufhin entweder per linkem Mausklick oder per Button in der Symbolleiste den semantischen Modus starten. Daraufhin öffnet sich ein Popup-Fenster über dem Text, in dem der User über ein Eingabeformular die gewünschten Informationen dieses Elements einträgt.

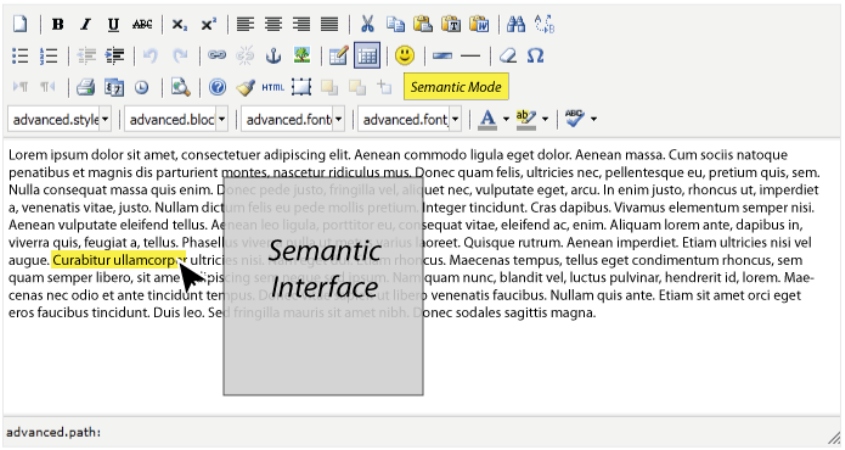


Abbildung 5.1: Konzeptentwurf: *Semantic RTE*.

Um Objekte bearbeiten zu können, wechselt man in den Bearbeitungsmodus des Editors und markiert das gewünschte Objekt. Objektvorschläge müssten in einer zusätzlichen Liste unter- oder oberhalb des Editors angezeigt werden.

5.1.2 Wordcloud

Die so genannte *Wordcloud* (siehe Abbildung 5.2) unterteilt den Vorgang des semantischen Anreicherns von Redaktionstexten strikt in zwei separate Arbeitsschritte. Visualisiert wird diese Tatsache durch einen Interface-Wechsel. Einerseits hat der User sein gewohntes Umfeld bei der normalen Texteingabe im Rich-Text-Editor von Typo3 – die Erstellung der Texte läuft damit genauso ab wie bisher.

Im zweiten Schritt wechselt der User dann in die semantische Ansicht der Erweiterung und findet dort keinen Texteditor, sondern eine *Wordcloud* vor. In dieser Ansicht kann der User den erstellten Text nicht bearbeiten – er wird, grafisch aufbereitet, in einzelnen Elementen angezeigt.

Object-Handling

Um ein Objekt auszuwählen, muss der User lediglich auf die gewünschten Elemente in der *Wordcloud* klicken. Hierbei gilt es zu beachten, dass ein User auch mehrere Wörter markieren können muss. Da die Objekte in diesem Entwurf allerdings nicht an eine RTE-Ausgabe gebunden sind, können jederzeit semantische Objekte als solche ausgegeben werden und so dem User übersichtlich darstellen, was bisher schon zugewiesen wurde.

Objekt-Vorschläge können bei dieser Ansicht auch problemlos in der *Wordcloud* grafisch hervorgehoben werden.

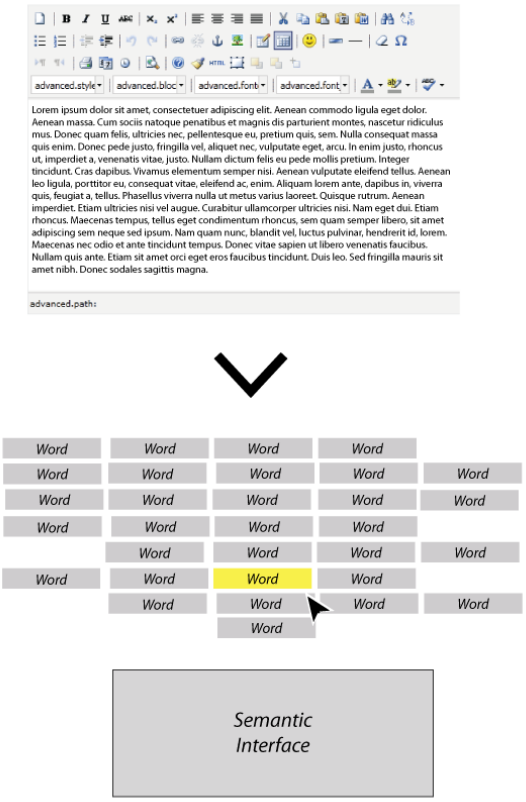


Abbildung 5.2: Konzeptentwurf: *Wordcloud*.

5.1.3 Semantic Graph

Die ersten zwei Entwürfe konzentrierten sich stark auf die Textrepräsentation sowie die Objektselektion, beim letzten Exemplar liegt der Fokus allerdings darauf, wie die semantischen Objekte schlussendlich eingetragen werden.

Die Texteingabe funktioniert weitgehend über den bekannten Rich-Text-Editor. Dieser wird nun allerdings durch ein grafisches Interface ergänzt, welches sich unter dem RTE befindet (siehe Abbildung 5.3).

Object-Handling

Die Objektauswahl funktioniert in diesem Fall wieder über die Textselektion im RTE. Dieser könnte aber, wenn gewünscht, auch durch eine *Wordcloud* ersetzt werden, wenn die Arbeitsschritte getrennt werden sollen. Sobald ein Objekt ausgewählt wurde, wird dieses in der grafischen Oberfläche als neues Element angezeigt. Nachdem der User die Klasse ausgewählt hat, wird diese symbolisch um das Objekt gelegt und zusammen bilden diese nun den Ob-

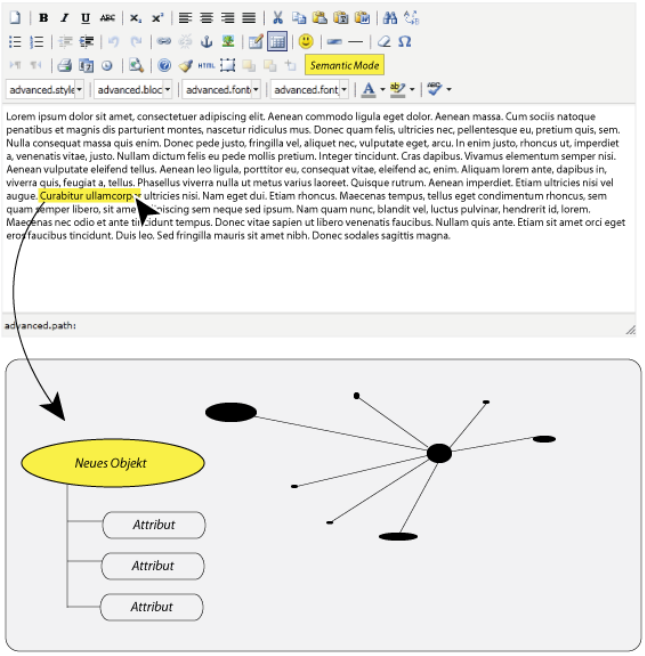


Abbildung 5.3: Konzeptentwurf: *Semantic Graph.*

jektkern. Der User kann neue Attribute anlegen, die daraufhin im Kreis um das Objekt angeordnet werden.

Schlussendlich kann der User über diese Darstellung einen semantischen Graphen anlegen und hat auch die Verknüpfungen, sofern solche existieren, sofort im Blick.

5.2 Diskussion

Im folgenden Abschnitt werden die vorgestellten Entwürfe miteinander verglichen und schlussendlich ein Resümee gezogen. Nun werden auch die technischen Umsetzungsmöglichkeiten miteinbezogen.

5.2.1 Vergleich der Entwürfe

Die drei Entwürfe liefern verschiedene Ansätze, wie das Interface schlussendlich gestaltet werden soll.

Die Macht der Gewohnheit

Ein großer Punkt, auf den es sich festzulegen gilt, ist die Tatsache, ob man sowohl die Texterstellung als auch die semantische Anreicherung in einer

einzigen Arbeitsumgebung belässt oder diese aufteilt. Ein großer Vorteil des *Semantic RTE*-Entwurfs ist dabei sicherlich die Tatsache, dass der User alles gesammelt in einem Interface vorfindet und sich durch das gewohnte RTE-Layout relativ schnell zurechtfinden wird. Fraglich bleibt jedoch, ob der User einen Überblick über den Status der semantischen Befüllung haben wird.

Im Rich-Text-Editor wird das Layout und die Struktur der Ausgabe gesteuert, es wird daher nicht möglich sein, am Text selber erkenntlich zu machen, dass es sich dabei um ein semantisches Objekt handelt, ohne den User vor die Frage zu stellen, ob es sich bei der Anzeige nun um ein semantisches Objekt oder eine Ausgabeformatierung handelt. Durch die problematische Hervorhebung ergeben sich auch weitere Probleme. So könnte es für den User schwer möglich sein, alte Objekte textlich und semantisch anzupassen, wenn beides über einen Editor gesteuert wird.

Technisch gesehen wird der User damit an eine festgesetzte Version des RTE gebunden und künftige Updates Seitens des RTE-Entwicklers können nicht mehr selbstständig ausgeführt werden, da die semantischen Funktionalitäten direkt daran gebunden sind. Folgedessen müssten diese Updates entweder ignoriert und damit dem User vorenthalten werden, oder durch extra Updates des semantischen Systems nachgeliefert werden.

Strikte Trennung

Einen anderen Ansatz bietet im Gegensatz dazu das *Wordcloud*-Konzept. Es existiert eine klare Trennung zwischen Texteingabe und semantischer Bearbeitung. Der User muss sich dafür allerdings an ein neues Teil-Interface gewöhnen. Die strikte Trennung zwischen diesen Arbeitsschritten erleichtert jedoch den allgemeinen Umgang mit dem Modul und stellt auch klar, wann der User den Text ändern kann und wo die semantische Bearbeitung stattfinden wird.

Durch die seperate Darstellung des Textes in einer *Wordcloud* kann das System hier objektorientierter arbeiten und dem Redakteur zu jeder Zeit übersichtlich anzeigen, welche Objekte schon mit semantischen Informationen ausgestattet sind und welche Vorschläge in der Datenbank gefunden wurden.

Die Eingabemöglichkeiten für neue Objekte sind in gewohnter Formularform gehalten. Bei der Umsetzung muss darauf geachtet werden, dass sich der User selbst im Formular nicht verliert und immer eine übersichtliche Strukturierung vorfindet.

Ein Großteil des Interfaces ist mit PHP und Javascript umzusetzen – bei der *Wordcloud* gibt es die Optionen, sie über Flash zu animieren oder normal als HTML Element zu erstellen. Flash im Typo3-Backend wäre jedoch nicht wirklich positiv, da es die Ladezeit der Inhaltselemente extrem verlangsamen würde und man mit einer Auslagerung in Flash die Tür für eine große Anzahl an Problemen öffnet.

Grafische Graphen

Das Konzept des *Semantic Graph* befasst sich hauptsächlich mit der semantischen Anreicherung und lässt die Texteingabe außen vor. Wichtig ist nur, dass die Objektauswahl mit dem Graphen verbunden werden kann - ob das nun über einen RTE oder auf andere Weise passiert, ist dabei nicht wirklich entscheidend. Der Ansatz, User mit grafischen Darstellungen und Animationen beim Aufbau der Wissensbasis zu unterstützen, ist nicht schlecht.

Hier könnte es allerdings bei der Umsetzung zu Schwierigkeiten kommen. Das vorgestellte Konzept lässt sich zufriedenstellend nur mit Flash umsetzen. Zuvor wurde schon festgestellt, dass es mit Flash-Applikationen im Typo3-Backend zu enormen Problemen kommen kann.

Davon abgesehen stellt sich die Frage, ob die grafische Darstellung der Graphen auf User nicht eher verwirrend und abschreckend wirkt.

5.2.2 Fazit

Die Wahl ist auf den Entwurf der *Wordcloud* gefallen. Das getrennte Interface verspricht eine übersichtliche, saubere Arbeitsfläche. User, die zuvor mit semantischer Anreicherung noch nicht vertraut waren, werden sich darin eher zurecht finden, als in einer grafischen Ansicht, die noch dazu durch die systemtechnischen Voraussetzungen sowieso kaum für Typo3 geeignet ist.

Kapitel 6

Semantic Object Editor

Im folgenden Kapitel wird im ersten Abschnitt auf die Systemarchitektur und anschließend auf die Umsetzung des Backend-Interfaces, der Frontend-Ausgabe sowie auch auf die der Named-Graph-Ausgabe des *Semantic Object Editors* eingegangen. Der Editor wurde im Rahmen des Masterprojekts als begleitendes Projekt entwickelt.

6.1 Systemarchitektur

6.1.1 Verwendete Technologien

Das Projekt wurde sowohl mit den allgemein im Web verbreiteten Programmiersprachen HTML, PHP und SQL als auch folgenden Technologien umgesetzt:

Typo 3

Der *Semantic Object Editor* (SOE) wurde im speziellen für das Content-Management-System *Typo 3* entwickelt. Ein grober Überblick zu Typo 3 ist in Abschnitt 4.1 zu finden. Die Herausforderung hierbei lag in dem Zusammenspiel zwischen Extension und der Typo3 Extension API. Es galt, sowohl in die interne Interfacedarstellung von Typo3 einzugreifen, als auch für die externen PHP-Klassen und Dateien eine Möglichkeit zu finden, Zugang zur Datenbank zu bekommen.

JQuery

Bei JQuery handelt es sich um eine Javascript-Klassenbibliothek, die frei im Web verfügbar ist. Im Gegensatz zu PHP läuft Javascript clientseitig ab – das bedeutet, dass der Code erst im Webbrowser des Users abgearbeitet wird. Laut einer Studie von W3Tech[1] ist JQuery die mit großem Abstand am wei-

[1] http://w3techs.com/technologies/overview/javascript_library/all

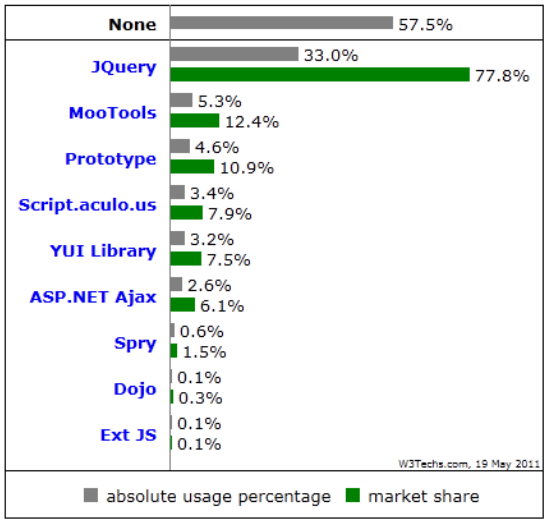


Abbildung 6.1: W3Tech Studie über die Verbreitung von Javascript-Bibliotheken.

testen verbreitete Javascript-Bibliothek (siehe Abbildung 6.1). Die bekanntesten Alternativen zu JQuery sind MooTools[2] und Prototype[3] – letzteres wird auch von Typo3 eingesetzt. Die aktuellste Version sowie eine detaillierte Dokumentation und Anwendungsbeispiele von JQuery findet man auf der offiziellen Homepage[4].

6.1.2 Typo3 Extension

Allgemeines

Einer der Gründe, warum sich Typo3 auf dem Markt durchsetzten konnte, ist die Möglichkeit, das System über sogenannte Extensions zu erweitern. Somit hat man als Programmierer die Möglichkeit, das System in großen Belangen seinen Vorstellungen anzupassen und es bietet der Community die Möglichkeit, das System um Module zu erweitern, die beim ursprünglichen Release der Versionen nicht enthalten waren oder noch nicht fertiggestellt wurden.

Wie man bei Abbildung 6.2 sehen kann, handelt es sich bei diesen Extensions um eigene Module, die je nach Bedürfnis variabel groß sein können. Über den sogenannten *Extension Manager* hat man die Möglichkeit, diese Erweiterungen zu verwalten, neue Extensions in das System zu importie-

[2] http://www.mootools.net/
[3] http://www.prototypejs.org/
[4] http://www.jquery.com/

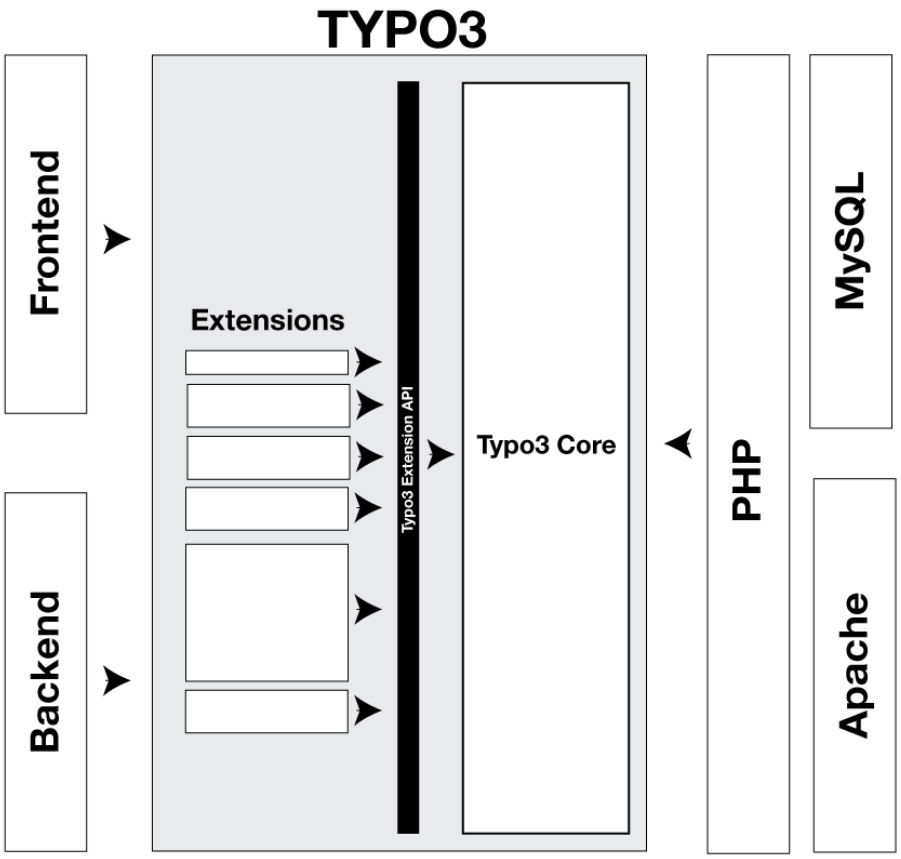


Abbildung 6.2: Zusammenspiel von Extensions und Systemkern von Typo3.

ren und zu installieren. In der offiziellen Typo3 Repository[5] findet sich eine große Anzahl von Extensions, die von der Community entwickelt und frei zur Verfügung gestellt wurden.

Um einen tieferen Einblick in die Entwicklung von Extensions zu bekommen, empfiehlt es sich, einen genaueren Blick in die *Typo3 Extension API* Dokumentation[6] zu werfen.

Der Extension-Manager bietet den sogenannten *Kickstarter*. Dieser gibt Usern die Möglichkeit, ohne viel Vorwissen einfache Erweiterungen für Typo3 anzulegen.

[5] http://typo3.org/extensions/repository/
[6] http://tinyurl.com/SB-Typo3ExtensionAPI

Extension-Setup

Auch für größere Projekte wie dem SOE kann man den *Kickstarter* nutzen. Typo3 verlangt eine in der Dokumentation genau spezifizierte Datenstruktur von Erweiterungen, damit diese problemlos mit der API kommunizieren können.

Durch den Start der Entwicklung über das systeminternen Erweiterungstool kann man sicherstellen, dass alle benötigten Systemstrukturen automatiersiert angelegt werden, was einem viel Zeit und Programmierarbeit eingesparren kann.

Der schnellste Weg, um mit einer Extension zu beginnen, führt daher direkt über den *Kickstarter*, in dem man sowohl sofort die benötigten Datenbankschnittstellen definieren kann als auch festlegt, um was für eine Extension es sich handelt – es gibt hier verschiedene Möglichkeiten wie die Backend- und Frontend-Extensions, um nur zwei Beispiele zu nennen.

6.1.3 Datenstruktur

Bis zu diesem Punkt wurden die Funktionsweisen der Texteingabe von Typo3 (siehe Abschnitt 4.1.2), sowie Datentriple zur Speicherung von semantischen Wissen (siehe Abschnitt 2.3.2) dargestellt. Mit diesen aufgebauten Erkenntnissen sollte es nun kein Problem mehr sein, eine Datenstruktur für die Erweiterung zu erarbeiten und zu verstehen.

Wie schon dargelegt wurde, wird der eingegebene Textblock von Typo3 in der Tabelle *tt_content* gespeichert. Da an der grundlegenden Architektur nichts verändert werden soll, wird auch der Fließtext, der über den Semantic Object Editor erstellt wird, weiterhin in dieser Tabelle abgespeichert werden. Um allerdings die Übersichtlichkeit und Struktur zu wahren, wird die Tabelle durch die Extension um ein weiteres Feld mit dem Namen *tx_SemanticObjectEditor* erweitert. Damit wird sichergestellt, dass auch Datensätze des Editors der Norm von gewöhnlichen Textelementen entsprechen und damit weiterhin Attribute wie *pid*, *crdate* und *sorting* aufweisen. Damit ist eine problemlose Datenverwaltung seitens Typo3 gesichert.

Die semantischen Daten werden hingegen in eigenen Tabellen abgespeichert. Hier musste ein Kompromiss zwischen puren semantischen Triple und programmtechnischer Verarbeitbarkeit gefunden werden, um mit den Daten auch in der Umgebung des Editors noch zufriedenstellend arbeiten zu können. Die Daten werden daher in vier Tabellen gespeichert (siehe Abbildung 6.3).

Anzumerken ist, dass nur wesentliche Datenstrukturen besprochen wurden. Typo3 besitzt natürlich noch weitere Tabellen, die aber für die Umsetzung des Projektes weitgehend irrelevant sind und daher keine weitere Erwähnung finden.

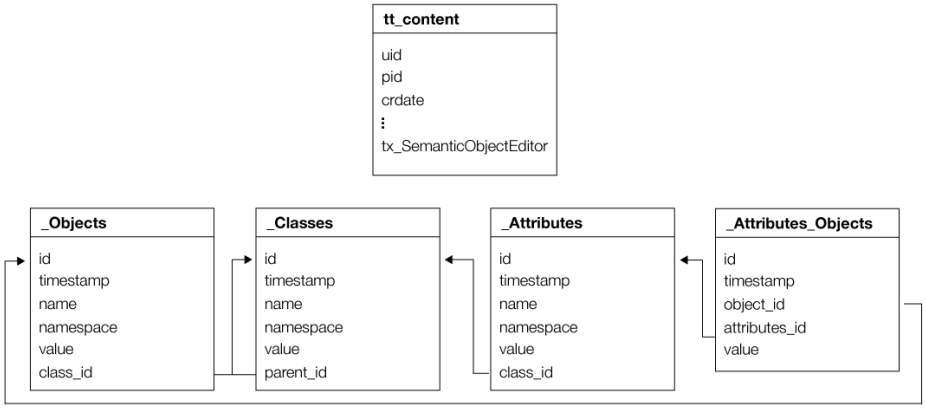


Abbildung 6.3: Datenmodell (*reduziert auf wesentliche Strukturen*).

6.2 Backend-Interface

6.2.1 Texteingabe-Interface

Wie bei der Anforderungsanalyse in Abschnitt 4.3.1 ausgeführt wurde, soll die für die User gewohnte Eingabemodalität beibehalten werden. Daraus folgt, dass der *Semantic Object Editor*, wie auch Typo3, ein Textfeld bietet, in das der Redakteur die Fließtexte eintragen kann. Außerdem wird, um den gewohnten Eingabekomfort zu bieten, auch mit einem Rich-Text-Editor[7] (RTE) gearbeitet.

Da der SOE im weiteren Vorlauf auf den verfassten Text des Redakteurs zugreifen muss, ist es nötig, dem User schon bei der Installation darauf hinzuweisen, dass der SOE nur mit einem bestimmten RTE funktioniert. Im Rahmen des Projektes wurde der *TinyRTE*[8] gewählt, da es sich dabei um einen stabilen Editor mit guten Anpassungsfunktionalitäten handelt.

Hat man diese Entscheidungen erstmal getroffen und eine Extension angelegt, die dem User die gewünschten Eingabeoptionen bietet, steht man aber zunächst vor einem neuen Problem. Die Struktur einer grundlegenden Typo3-Extension sieht vor, dass im Code vorgegeben wird, welche Eingabefelder und Datenbanktabellen benötigt werden und wie diese zusammenspielen.

Die weitere Datenverwaltung über diese vordefinierten Felder läuft danach allerdings wieder über das interne System von Typo3. Es ist daher nötig, sich über die sogenannten *UserFunc*-Funktionalität der Extension-API Zu-

[7]Rich-Text-Editoren sind im Webbereich Programme zum Erstellen, Bearbeiten und Formatieren von Texten, ohne dafür Programmierkenntnisse haben zu müssen.

[8]Der TinyRTE ist im Typo3-Repository frei verfügbar: http://typo3.org/extensions/repository/view/tinyrte/current/

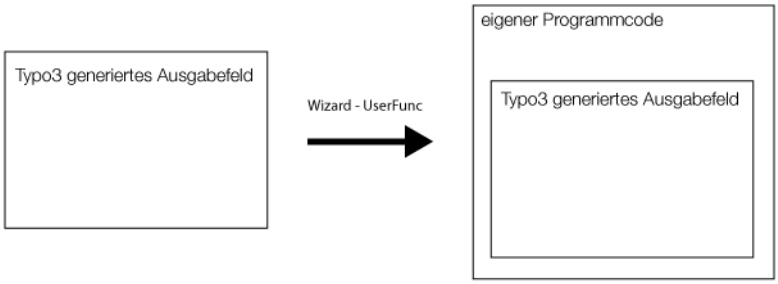


Abbildung 6.4: Illustration der Userfunc-Funktion.

griff auf das Ausgabeinterface zu verschaffen. Dies funktioniert über folgende Ergänzungen in der Datei *ext_tables.php* der Extension:

```
1 'wizards' => array(
2   'uproc' => array(
3     'type' => 'userFunc',
4     'userFunc' =>'EXT:SemanticObjectEditor/src/startObjectEditor.php:
      user_class->initiateInterface',
5     'params' => array(
6       'wrapTag' => 'u'
7       ...
```

Der Eingriff über die *UserFunc* erlaubt es der Erweiterung, den kompletten Programmcode, der von Typo3 für die Ausgabe der Formulare erzeugt wurde, durch eine eigene Funktion zu schleifen und damit den Ausgabecode zu manipulieren. Dadurch wandert die Verantwortung über das Userinterface zur eigenen Extension (siehe Abbildung 6.4), und genau das wird benötigt, um dem Redakteur ein eigenes Ausgabeinterface bereitstellen zu können.

6.2.2 Semantisches Anreichern

Im nächsten Schritt muss der SOE den zuvor erstellten Fließtext mit semantischen Informationen anreichern.

Textanalyse und Wordcloud

Sobald der Redakteur mit dem Erstellen des Textes fertig ist, wechselt er in den sogenannten *Semantic*-Modus des SOE. Daraufhin verändert sich das Interface – weg von der Text-Ansicht des RTE hin zur semantischen Oberfläche. Es wird nun vorerst auf den RTE zugegriffen und der erstellte Text ausgelesen. Dieser Schritt ist auch der Grund, warum, wie zuvor erwähnt, die Extension genau wissen muss, mit welchem RTE der Redakteur arbeitet.

Es gibt zwar ein grundsätzliches Input-Feld von Typo3, in dem der Text nach einmaligem Speichern abrufbar wird, allerdings müsste man dann vom

```
1 postVars.input=getTinyMCEInput();
2 $j('#testContainer').load(getUrl()+'wordInterface.inc.php',postVars,
      finishedStep1);
3
4 //calls the content of the tinyMCE Element
5 function getTinyMCEInput(){
6   var inputCall=frames[0].document.getElementById('tinymce').innerHTML;
7   return inputCall;
8 }
```

Programm 6.1: Codeauszug aus JavaFuns.js.

User vor jedem weiteren Schritt immer verlangen, dass er das Datenelement zwischen Textänderungen und semantischen Anpassungen abspeichert. Dies wäre nicht nur für die Userbility sehr unvorteilhaft, sondern auch für den sicheren Programmablauf ein unüberwindbares Hindernis. Ohne genauen Zugriff auf den RTE kann das Programm nie genau wissen, ob der User seinen bereitgestellten Inhalt schon ordnungsgemäß gespeichert hat und damit würde man mit Änderungen an der Semantik alle zuvor vorgenommenen Änderungen wieder überschreiben.

Die Lösung dafür war, sich auf den tinyRTE festzulegen. Damit ist garantiert, dass die Extension immer Zugriff auf den aktuellen Text des Redakteurs hat und man läuft unter keinen Umständen Gefahr, vom User erstellte Inhalte zu überschreiben.

Programm 6.1 zeigt, wie in Zeile 6 auf den iFrame des tinyRTE zugegriffen wird und direkt über die innerHTML-Funktion der HTML-Code des erstellten Textes ausgelesen wird. In Zeile 2 wird dann unter anderem der eingelesene Text über die *jQuery load*-Funktion an die *wordInterface.inc.php* weitergegeben, worin der ausgelesene Inhalt weiter verarbeitet wird.

Daraufhin wird eine sogenannte *Wordcloud* angelegt, die den bereitgestellten HTML-Code erst einmal bereinigt. Dabei ist allerdings zu beachten, dass die Elemente nicht einfach gelöscht werden können – immerhin sollen ja die Formatierungen des Users nicht komplett verloren gehen. Weiterhin könnten sich in dem Text auch schon semantische Objekt-Marker befinden, die mit einem reinen HTML-Cleaning komplett verloren gehen würden.

Der angewandte Lösungsansatz sieht nun wie folgt aus (siehe Programm 6.2): Zuerst wird der Text mit so genannten *Regular Expressions*[9] nach semantischen Objekt-Platzhaltern durchsucht – diese repräsentieren bereits mit semantischen Informationen ausgestattete Terme (siehe Abschnitt 6.3.1). Dabei ist speziell durch die große Variabilität der Dummy-Elemente ein komplexer Ausdruck nötig, um wirklich alle Elemente zu finden.

[9]Bei Regular Expressions handelt es sich um eine bestimmte Zeichenkette, die nach syntaktischen Regeln einen bestimmten Ausdruck beschreiben. Für nähere Informationen siehe http://de.wikipedia.org/wiki/Regulärer_Ausdruck

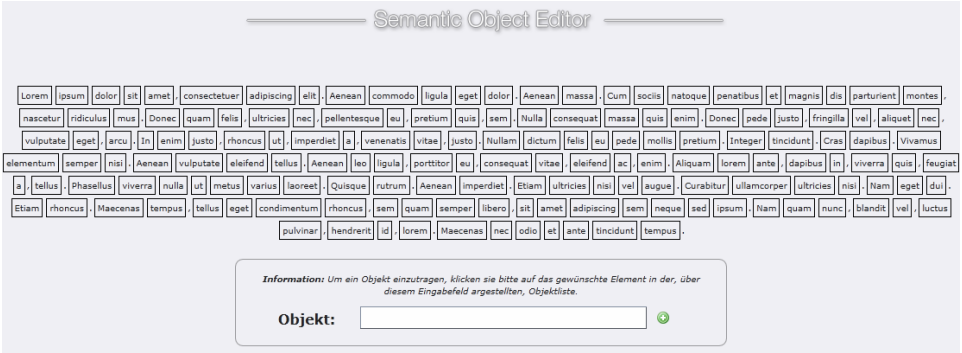


Abbildung 6.5: Beispiel einer neuen Wordcloud, befüllt mit Blindtext.

Sobald ein Dummy-Element gefunden wird, wird eine Callback-Funktion aufgerufen (siehe Zeile 18), in der das komplette gefundene Element in einer Liste abgespeichert und durch ein einfaches Flag-Element[10] ersetzt, das zur Strukturierung noch die Indexzahl des Elementes in der Liste trägt.

Selbige Prozedur führt dann auch dazu, dass jegliche HTML-Elemente durch Flags ersetzt und im Hintergrund in einer eigenen Liste gesichert werden.

Eine Differenzierung in zwei Arbeitsschritte ist in diesem Fall unabdingbar, da es für die Darstellung der Wordcloud nötig ist, zwischen HTML und SemanitcObject-Flags problemlos unterscheiden zu können. Erstere können für die Cloud komplett ignoriert werden, während zweitere als bereits erkannte semantische Objekte aufgelistet werden müssen.

Abschließend wird der Inputtext noch Wort für Wort in eine Liste aufgeteilt, um mit der grafischen Darstellung der Wordcloud beginnen zu können. Dafür wird jedes Listen-Element, mit Ausnahme von Satzzeichen, als *Span*-Element grafisch aufbereitet. Die Länge des Textes ist dabei nicht relevant, da die *Wordcloud* keine Begrenzungen hat und der User bei längeren Texten im schlimmsten Fall einfach nur scrollen muss. In Abbildung 6.5 ist eine Worldcloud zu sehen, die mit Blindtext gefüllt wurde, bei der keine bekannten Elemente gefunden wurden und bisher noch keine neuen Elemente definiert wurden.

Vorschlagen von Objekten

Wenn der Redakteur nun bei jedem neuen Fließtext alle semantischen Objekte bei jedem Vorgang neu anlegen müsste, würde das den User nicht nur

[10] Bei einem Flag-Element handelt es sich um einen zuvor festgelegten Ausdruck, der für das Programm immer dieses spezielle Element repräsentiert. Im Falle der semantischen Dummy Elemente ist dieses Flag-Element als *SemanticObjectContainer* defininiert. Dieser wird im Quellcode noch mit dem benötigten Index ergänzt. Zum Beispiel: *SemanticObjectContainer_ 1*

```
1 function organizeInput($content){
2     //semantische objekte ausnehmen
3     $re1='(<)'; # Any Single Character 1
4     $re2='(span)';  # Variable Name 1
5     $re3='(\\s+)';  # White Space 1
6     $re4='(id)';  # Variable Name 2
7     $re5='(=)'; # Any Single Character 2
8     $re6='(")'; # Any Single Character 3
9     $re7='(SemanticObjectId)';  # Word 1
10    $re8='(_)'; # Any Single Character 4
11    $re9='(\\d+)';  # Integer Number 1
12    $re10='(")';  # Any Single Character 5
13    $re11='(>)';  # Any Single Character 6
14    $re12='((?:[a-z][a-z]+))';  # Word 2
15    $re13='.*?';  # Non-greedy match on filler
16    $re14='(<\\/span>)';  # Tag 1
17    $re="/".$re1.$re2.$re3.$re4.$re5.$re6.$re7.$re8.$re9.$re10.$re11.
      $re12.$re13.$re14."/is";
18    $content = preg_replace_callback($re, array($this,'objectCallback'),
       $content);
19    //html ausnehmen
20    $content = preg_replace_callback("/<(.|\n)+?>/", array($this,'
      callback'), $content);
21    return $content;
22  }
23
24  /**
25   * Html Callback, sets the HTML Marker
26   */
27  function callback($match){
28        $this->htmlList->append($match);
29    return " ".wordcloud::htmlFlag."_".$this->htmlList->count()." ";
30    }
31
32    /**
33     * Object Callback, sets the SemanticObject Marker
34     */
35    function objectCallback($match){
36      $this->objectDummyList->append($match);
37    return " ".wordcloud::objFlag."_".$this->objectDummyList->count()."
      ";
38    }
```

Programm 6.2: Bereinigung des HTML Inputs.

relativ schnell dazu bringen, genervt aufzugeben, sondern auch dazu, dass eine große Anzahl an Duplikaten in der Wissensbasis entstehen würden.

Um dieses Problem zu lösen, werden die einzelnen Textelemente mit der bereits existierenden Datenbank abgeglichen. Dabei wird für jedes Wort untersucht, ob es mit einem bereits existierenden semantischen Objekt in Ver-

bindung steht. Die Identifizerung läuft dabei über die Namen oder die Werte der semantischen Objekte selbst.

```
1 $objectQuery=$this->dbconnect->mysql_query("SELECT * from
      tx_semanticObjectEditor_Objects WHERE name LIKE '%".$value."%' OR
      value LIKE '%".$value."%'");
```

In der ursprünglichen Planung der Vorschlagfunktion war außerdem noch vorgesehen, dass die gelieferten Worte auch mit den Attributen der Objekte verglichen hätten werden sollen, was zwar technisch kein Problem gewesen wäre, der Programmlogik allerdings nicht entsprochen hätte. Dem User werden in der Wordcloud die gefundenen Objekte direkt angezeigt; auch auf der Webseite werden die Objekte ausgebenen. Daher macht es Sinn, die Datenbank nach diesen Objekten zu durchsuchen und dem User die Möglichkeit zu geben, diese auszuwählen. Wenn im Text allerdings Werte vorkommen, die bei bestimmten semantischen Objekten als Attribute eingetragen wurden, diese Objekte selbst aber gar nicht im Text vorkommen, wäre es nicht zielführend, dem User diese Objekte vorzuschlagen, da sie im Text selbst überhaupt nicht auftauchen.

Eine Untersuchung der möglichen Attribute wäre nur für die bereits gefundenen Objekte sinnvoll, um dem Redakteur die vorgeschlagenen Objekte nach einem Prioritätenwert zu sortieren.

Vorgeschlagenes Element auswählen

Nachdem das System Übereinstimmungen mit semantischen Objekten feststellen konnte, wird dem User direkt in der Wordcloud bei den konvergenten Elementen ein vorliegender Vorschlag angezeigt. Wie in Abbildung 6.6 nachvollziehbar dargestellt, kann der User daraufhin per Klick auf eines der Elemente ein Dialogfenster öffnen. Darin findet er eine Liste vorgeschlagener Elemente und kann sich, wenn nötig, auch eine Detailansicht aller zugewiesenen Attribute ansehen.

Um den Vorgang erfolgreich abzuschließen, muss das gewünschte Objekt nur ausgewählt und per Klick bestätigt werden. Daraufhin wird das Objekt sowohl in der Wordcloud als auch im Blindtext eingetragen. Übertragen werden diese Daten im Hintergrund wieder über die AJAX-Funktion von jQuery, der User muss also die Seite nicht selbst aktualisieren, um die Änderungen auch optisch wahrnehmen zu können.

Die Darstellung wird weitgehend über CSS gestaltet. Vorgeschlagene Elemente werden dafür extra in einen *span*-Container mit der Klasse *suggestionBox* gepackt, damit auch Objekte, die aus mehreren Wörtern bestehen, problemlos dargestellt werden können. Ähnlich wird auch mit den bereits definierten Objekten gearbeitet – diesen wird ein *span*-Container mit der Klasse *specifiedObjects* zugewiesen.

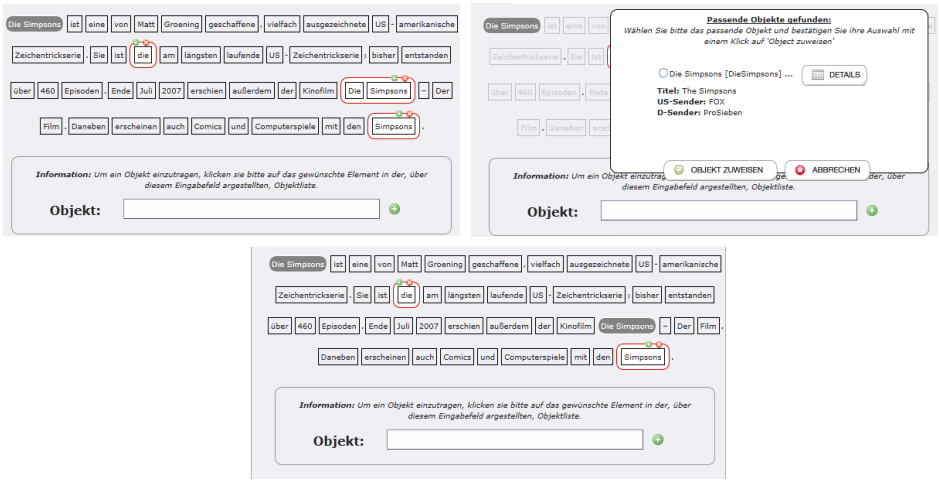


Abbildung 6.6: Beispiel für die Auswahl eines vorgeschlagenen Objektes.

Neue Objekte anlegen

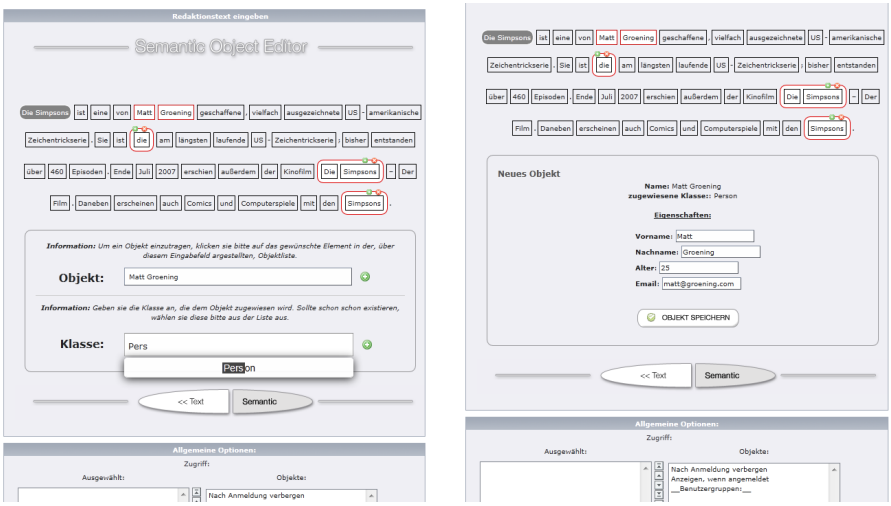
Damit der *Semantic Object Editor* überhaupt Objektvorschläge erarbeiten kann, müssen vom Redakteur erstmal Objekte angelegt werden. Dies bringt logischerweise zu Beginn einen erhöhten Arbeitsaufwand mit sich – immerhin wird das System mangels Daten kaum Vorschläge liefern. Da sich aber Webseiten, seien es nun reine Firmenauftritte oder Themenplattformen, meist mit einem abgegrenzten Themenkreis befassen und die semantischen Objekte eher repetitiv sind, lässt sich der Rückschluss ziehen, dass der Arbeitsaufwand schnell sinken wird.

Abbildung 6.7 zeigt in zwei Schritten, wie der User üblicherweise neue Objekte anlegen kann. In Schritt (a) werden die Wörter per Mausklick markiert, die künftig das semantische Objekt bilden sollen – im Sinne der Übersichtlichkeit werden im selben Moment die gewählten Ausdrücke auch in der Eingabeleiste bei *Objekt* gelistet.

Nach der Bestätigung über den grünen *Plus-Button* wird im nächsten Schritt das Klassenzuweisungsinterface geöffnet. Da es sich in diesem Beispiel bei *Matt Groening* um eine Person handelt, ist es auch ratsam, diese Klasse zuzuweisen. Sobald der Redakteur den Klassennamen eingibt, wird im Hintergrund in der Datenbank nach gleichlautenden Klassennamen gesucht und bei möglichen Übereinstimmungen vorgeschlagen. Zur Umsetzung dieser *Autosuggest-Funktion* wurde ein von Drew Wilson für jQuery entwickeltes PlugIn[11] gewählt und auf die Bedürfnisse der Erweiterung angepasst.

Bei Abbildung 6.7 Schritt (b) werden daraufhin dem User die dieser Klasse zugewiesenen Eigenschaften angezeigt. Er hat nun die Möglichkeit, diese

[11] Weitere Informationen und ein Download des PlugIns: http://code.drewwilson.com/entry/autosuggest-jquery-plugin



(a) (b)

Abbildung 6.7: Illustriert den typischen Vorgang des Anlegens eines neuen semantischen Objektes.

mit weiteren Daten zu füllen, die später als semantische Informationen bei diesem Objekt angezeigt werden sollen.

Nachdem der Vorgang abgeschlossen wurde, werden auch hier die semantischen Daten wieder über AJAX direkt an die Datenbank übertragen und abgespeichert, ohne dass der User sich weitere Gedanken darüber machen muss.

Neue Klassen anlegen

Bei Abbildung 6.7 wurde allerdings vorausgesetzt, dass die semantische Klasse, die dem neuen Objekt zugewiesen werden soll, schon in der Datenbank existiert. Es muss natürlich auch möglich sein, diese Klassen selbst anzulegen und zu definieren.

Wie in Abbildung 6.8 zu sehen, funktioniert dieser Vorgang relativ einfach und unproblematisch. Sollte der Redakteur keine Klasse aus der Datenbank vorgeschlagen bekommen, kann er diese über einen Klick auf den zugehörigen *Plus-Button* anlegen.

Im darauf folgenden Interface kann man der Klasse die benötigten Attribute zuweisen – die fünf Felder, die zu Beginn angezeigt werden, können beliebig erweitert werden.

Nachdem der Redakteur die Eingaben bestätigt hat, wechselt er wieder zu dem aus Abbildung 6.7 bekannten Schritt (b), in dem die Attributwerte dem jeweiligen Objekt zugeordnet werden.

Abbildung 6.8: Illustriert den typischen Vorgang des Anlegens einer neuen semantischen Klasse samt Attribute.

Funktionale Kompromisse für Userbility

Um den Arbeitsschritt der semantischen Informationsanreicherung für Redakteure so verständlich wie möglich zu gestalten, waren auch bei diesem Vorgang Kompromisse nötig. Wie in der Anforderungsanalyse in Abschnitt 4.3.2 erläutert, ist es wichtig, einen Mittelweg zwischen semantischer Anreicherungsfunktionalität und Userbility zu finden. Die Zielgruppe des *Semantic Object Editors* sind normale Typo3-Redakteure, die bisher kaum Erfahrung mit semantischer Anreicherung sammeln konnten.

Daher wurde die Entscheidung getroffen, auf bestimmte Funktionen ganz bewusst zu verzichten. So ist es nicht möglich, dass der User selbst die Typen der semantischen Datensätze festlegen kann. Es ist auch nicht gedacht, dass User Objekte als Attribute anderer Objekte eintragen. Auch diese Funktionalitäten wären für die technische Umsetzung kein Problem gewesen, hätten aber eine große Fehlerquelle dargestellt, die für neue User hauptsächlich zu Verwirrungen geführt hätten. Mit dem gewählten Kompromiss ist es nun zwar nicht möglich, eine hochkomplexe Wissenbasis zu kreieren - aber hier muss man ganz einfach festhalten, dass die Erweiterung dafür auch nicht gedacht ist und man dafür vermutlich generell zu einer anderen technologischen Grundlage als Typo3 greifen sollte (siehe Alternativen in Kapitel 3).

6.2.3 Import von externem Vokabular

Im Laufe der Arbeit wurde festgestellt, dass der Import von externen Vokabeln für eine semantische Anreicherung von erheblichem Vorteil ist. Es wurde daher eine mögliche Schnittstelle im Programm integriert, die es dem User theoretisch ermöglicht, externe Vokabeln in das System zu integrieren.

Da der Fokus der Entwicklung allerdings auf der Ausarbeitung eines Userinterfaces für die semantische Anreicherung in Typo3 gelegt wurde und es sich bei dieser Erweiterung um einen Prototypen handelt, wurde die Import-Funktion nicht vollkommen ausprogrammiert.

6.3 Frontend-Ausgabe

Der folgende Abschnitt befasst sich mit der Ausgabe des *Semantic Object Editors* im Frontend der Seite.

6.3.1 SemanticObject Dummy-Elemente

Durch die strukturellen Gegebenheiten und Anforderungen von Typo3 wurde die Datenverwaltung in zwei Teile gespalten: einerseits in die normale Datenverwaltung der Typo3 Inhaltselemente über die Tabelle *tt_content* und andererseits die Wissensdatenbank in den eigenen Tabellen (siehe Abschnitt 6.1.3). Um trotzdem eine Verbindung zwischen diesen Datensätzen herzustellen, wurden so genannte Platzhalter-Elemente eingeführt, die im Fließtext an den Stellen eingebunden werden, an denen semantische Objekte zugewiesen wurden. In folgendem Codebeispiel ist die Anwendung eines Dummy-Elements ersichtlich[12]:

```
1 <b><span id='SemanticObjectId\_1'>Die Simpsons</span></b> ist eine...
```

Die angeführte Object-Id ist eindeutig und repräsentiert die in der Datenbank zugewiesene ID des Datensatzes.

Sobald im Frontend eine Seite geladen wird, die Inhaltselemente des *Semantic Object Editors* enthält, wird die *tx_SemanticObjectEditor_pi1* Klasse der Extension aufgerufen, in der über die *Main-Funktion* die Ausgabe der Erweiterung gesteuert wird.

```
1 function main($content,$conf){
2   $text=$this->ext_getContentVar("tx_SemanticObjectEditor");
3   $htmlcontent=$this->pi_RTEcssText($text);
4   $cw=new wordcloud($htmlcontent);
5   $content=$cw->getRDFaOutput();
6   return $content;
7 }
```

[12]Das verwendete Textbeispiel stammt von folgender Seite: http://de.wikipedia.org/wiki/Die_Simpsons

Darin wird vorerst über die von Typo3 gelieferten Funktionen der benötigte Inhalt aus der *tt_content* Tabelle geladen (siehe Zeile 2) und mit den HTML Tags angereichert, die von Typo3 beim Abspeichern des Datensatzes maskiert wurden (siehe Zeile 3). Im letzten Schritt wird dann die schon bekannte *Wordcloud*–Klasse erzeugt, über die der HTML-Code nach Dummy-Elementen durchsucht wird.

```
1 const objFlag = 'SemanticObjectContainer';
2 function isObjContainer($value){
3   $parts=explode("_", $value);
4   if($parts[0]==wordcloud::objFlag){
5     return true;
6   } else {
7     return false;
8   }
9 }
```

Sobald das System ein Dummy-Element erkannt hat, wird dieses an die RDFa-Darstellungsfunktion weitergegeben, die im folgenden Abschnitt besprochen wird. Nach dem erfolgreichen Abschluss des Vorgangs setzt die Erweiterung den fertigen HTML-Code wieder zusammen und übergibt es an die Typo3 Extension API, die die restliche Darstellung wieder übernimmt.

6.3.2 RDFa Darstellung in HTML

Um den mit semantischen Informationen angereicherten Inhalt auf der Seite auszugeben, werden die semantischen Objekte mit RDFa (siehe Abschnitt 2.2) auf der Seite ausgegeben. Folgende Codezeilen zeigen, wie der HTML-Code für jedes einzelne Objekt berechnet wird.

```
1 //Object with Class
2 $output.='<span typeof="'.$this->getNameSpaceAlias($row['cnamespace'])."
      :".$row['cname'].'" about="'.$this->getNameSpaceAlias($row['
      onamespace'])."::".$row['oname'].'">'.$row['ovalue'];
3 ...
4 //each Attribute
5 $output.='<span property="'.$this->getNameSpaceAlias($row2['namespace'])
      .":".$row2['name'].'" content="'.$row2['value'].'"></span>';
6 ...
```

Dabei wird das Objekt über die eindeutige ID geladen, die über das Dummy-Element im normalen HTML-Code gespeichert war. Zusätzlich werden auch die zugehörigen Klassen- und Attributinformationen aus der Datenbank geladen und in das vorgebene RDFa-Gerüst eingebaut.

Die sogenannten Namensräume werden extra über die *getNameSpaciaAlias* Funktion bearbeitet. Die Idee dahinter ist, dass man Instanzen, die sich ständig wiederholen, durch Präfixes zusammenfasst und damit einen übersichtlicheren Code generiert [19, S. 41]. Das System sammelt somit intern alle vorkommenden Namensräume in einer Liste zusammen und vergibt danach passende Präfixes. Dabei ist es irrelevant, ob der Namensraum in der

Abbildung 6.9: Zeigt die optische Darstellung sowie einen Teil des zugehörigen HTML Code mit RDFa Elementen.

eigenen Domäne liegt oder von externen Quelle bezogen wurde. Sobald alle Objekte berechnet sind, wird der zugehörige RDFa-Code an die Ausgabe zurück gegeben und der Dummy-Platzhalter ersetzt (siehe Abbildung 6.9).

6.4 Named-Graph-Darstellung

Dieser Abschnitt behandelt die so genannte Named-Graph-Darstellung. Diese wird angezeigt, sobald die eindeutige URI eines vom Redakteur selbst erstellten semantischen Elementes als URL angesprochen wird.

6.4.1 URI festlegen

Die Festlegung der URI ist nur für Elemente wichtig, die vom Redakteur in dieser Extension angelegt wurden. Jegliche externe Elemente sind dabei irrelevant, da diese sowieso mit einer eigenen URI versehen sind.

```
1 $namespace="http://".$_SERVER['SERVER_NAME'].str_replace("typo3conf/ext/
      SemanticObjectEditor/src/wordInterface.inc.php", "", $_SERVER['
      REQUEST_URI'])."onthology/";
2 $objectNamespace=$namespace."object/";
3 $classNamespace=$namespace."class/";
4 $attributeNamespace=$namespace."attribute/";
```

Die URI setzt sich aus dem Namensraum sowie dem Namen des Objektes, der Klasse oder des Attributes zusammen. Jedes Element dieser drei Typen

ist durch eine eigenständige URI eindeutig im System ansprechbar – dies wird beim Anlegen der Elemente sichergestellt.

Den Namensraum sucht sich der *Semantic Object Editor* selbstständig aus den Serverdaten des laufenden Typo3-Systems.

Läuft das Typo3-System also zum Beispiel unter der Domäne

www.beispieladresse.at,

ist der berechnete Namensraum für die das Objekt „Beispielobjekt“:

www.beispieladresse.at/onthology/object/Beispielobjekt.

Das System ordnet dabei den drei Typen Objekt, Klasse oder Attribut jeweils einen eigenen Namensraum zu[13]. Dies soll einerseits zu einem übersichtlicheren Namensraummanagement führen, stellt andererseits auch sicher, dass es im System zwar beispielsweise Objekte und Klassen mit dem gleichen Namen geben darf, diese aber trotzdem weiterhin eindeutige URIs aufweisen.

6.4.2 URI als URL

Um die eindeutige URI eines semantischen Objekts schlussendlich auch als URL ansprechbar zu machen, bedarf es einer kleinen Anpassung in der .htaccess-Datei des entsprechenden Servers. Über diese Datei können verzeichnisspezifische Änderungen am Webserver vorgenommen werden – wie zum Beispiel in diesem Fall die Anforderungen, unter welchen Bedingungen bestimmte Pfade ansprechbar sind[14].

```
1 RewriteRule onthology/.* typo3conf/ext/SemanticObjectEditor/src/
      onthology.php [L]
2 RewriteRule .*\.(html|pdf)$ index.php [L]
3 #RewriteRule .* index.php [L] /++/
```

In Zeile 1 wird der Server darauf hingewiesen, dass künftig alle Anfragen, die über die URL serveradresse/onthology ankommen, zu der *onthology.php* Datei der Erweiterung weitergereicht werden sollen.

Die neue Regel in Zeile 2 besagt, dass alle normalen Anfragen mit HTML- oder PDF-Endung weiterhin zur gewohnten *index.php* von Typo3 geleitet werden sollen.

Damit die neuen Regeln aber überhaupt funktionieren, muss erst die alte Regel in Zeile 3 auskommentiert werden – immerhin besagt diese, dass alle Anfragen zur *index.php* von Typo3 weitergeleitet werden sollen und würde damit die zuvor eingefügten Regeln negieren.

Die Anpassung dieser Datei könnte einfache Redakteure, die mit normaler Internetprogrammierung nichts zu tun haben, etwas überfordern. Der Vorgang selbst ist zwar nicht kompliziert und im Userhandbuch genau beschrie-

[13] Im angeführten Beispiel ist dies /object/ für ein semantisches Objekt. Klassen bekommen hingegen /class/ und Attribute /attribute/ zugeteilt.

[14] Weitere Informationen zu .htaccess: http://de.wikipedia.org/wiki/Htaccess

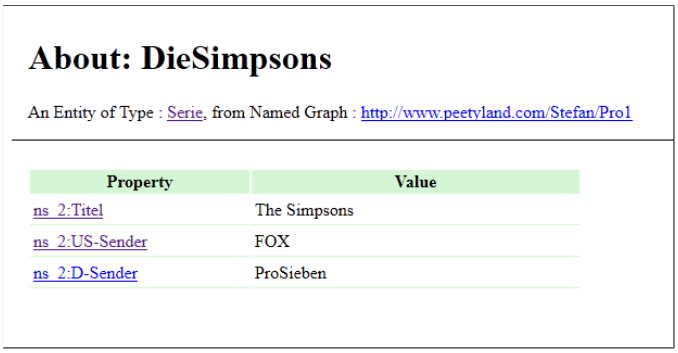


Abbildung 6.10: Screenshot der Named-Graph Ausgabe.

ben – es muss trotzdem damit gerechnet werden, dass nicht jeder Anwender selbst diese Dateien ändern möchte. Es sollte daher angemerkt werden, dass zwar ohne diese Anpassung die vom System erstellen URI's nicht per URL ansprechbar werden, wie aber in Abschnitt 4.3.4 festgestellt wurde, ist diese Tatsache auch keine zwingende Vorgabe [29, S. 66]. Die Extension selbst funktioniert natürlich auch ohne dieses Teilmodul.

6.4.3 Named-Graph-Ausgabe

Die Named-Graph-Ausgabe wird schlussendlich angezeigt, wenn die URI als URL angesprochen wird. Wie auch schon in der Anforderungsanalyse besprochen, wurde sich bei der Umsetzung der Ausgabe an DBpedia orientiert. Anzumerken ist außerdem, dass auch hier alle Informationen zusätzlich zur optischen Ausgabe auch per RDFa im Code zur Verfügung gestellt werden. Außerdem hat der User an dieser Stelle die Möglichkeit, sich durch die verlinkten Elemente durchzuklicken, um weitere elementspezifische Informationen zu erhalten (siehe Abbildung 6.10).

Kapitel 7

Diskussion

7.1 Einführung

Im Rahmen des Thesis-Projekts wurde der *Semantic Object Editor* entwickelt (siehe Kapitel 6). Ziel dieser Diskussion ist es nun, die Ergebnisse genauer zu betrachten und zu hinterfragen, ob und wie Ansätze von existierenden Projekten (siehe Kapitel 3) integriert wurden, ob diese verbessert werden konnten, womit es Probleme gab und wo noch Verbesserungsansätze gefunden wurden.

Dabei wäre es natürlich nicht zielführend, das eigene Projekt nur mit positiven Vergleichen zu loben oder es durch unrealistische Anforderungen in ein negatives Licht zu rücken – es gilt vielmehr, das gesamte Projekt unter Berücksichtigung der gewonnenen Erfahrungen aus Recherche und Projektarbeit nocheinmal zu reflektieren und daraus Rückschlüsse auf weitere Lösungswege zu ziehen.

7.2 Evaluierung des Semantic Object Editors

7.2.1 Systemintegration

Bei der Umsetzung der Applikation wurde sich an den Entwicklungsrichtlinien für Systemerweiterungen des Typo3-Kernsystems orientiert, es sollte daher zu keinen Komplikationen bei der Integration kommen. User, die bereits Erfahrung mit dem Import von Extensions sammeln konnten, werden hier keinerlei Probleme haben.

Die normale Texteingabe von Typo3 wird dabei nicht beeinflusst. Es besteht daher auch nicht die Gefahr, dass User mit einer späteren Integration des Editors ins System in Gefahr laufen, bisher erstellte Inhalte zu überschreiben oder unbrauchbar zu machen. Der Editor ist nach der Installation für Redakteure als Plugin auswählbar. Bei unerfahrenen Anwendern könnte eine geringe Gefahr bestehen, dass sie das Modul nach der Installation nicht

sofort finden – aber spätestens nach einem Blick in das Handbuch sollte dieses Problem beseitigt sein.

Eine größere Barriere könnte hingegen die Anpassung der *htAccess*-Datei darstellen. Einerseits gibt es die Möglichkeit, dass Redakteure keinen Serverzugang besitzen, andererseits könnte die Anpassung dieser Datei für unerfahrene Anwender trotz der genauen Anleitung im Benutzerhandbuch problematisch bleiben. Auf den ersten Punkt hat man von der Entwicklerseite keinen Einfluss – hier sollte man dem User zumindest empfehlen, sich an die zuständigen Systemadministratoren zu wenden. Um dem User einen direkten Kontakt mit Programmcode zu ersparen, wäre es außerdem hilfreich, eine vorbereitete *htAccess*-Datei in das Installationspaket zu integrieren – damit muss die Datei nicht angepasst, sondern nur überspielt werden.

7.2.2 Interface

Texteingabe

Nachdem bei der Entwicklung die Entscheidung getroffen wurde, bei der Eingabemöglichkeit auf den Entwurf eines getrennten Eingabe- und Semantic-Interface mit Wordcloud zu setzen (siehe Kapitel 5), hat sich für den Typo3-Redakteur an der gewohnten Texteingabe nichts geändert. Einzig die Voraussetzung mit dem tinyRTE zu arbeiten, könnte für User, die diesen Rich-Text-Editor bisher noch nicht integriert hatten, eine kurze Umgewöhnungsphase nach sich ziehen.

Beim *One Click Annotator* dient der tinyRTE als Systemgrundlage – dies ist beim *Semantic Object Editor* allerdings nicht der Fall. Hier dient dieser als Schnittstelle, um an die vom User erstellten Inhalte zu kommen und diese nach der Annotierung wieder einspielen zu können. Um den User nicht fix an den tinyRTE als Eingabesystem binden zu müssen, wäre vorstellbar, in künftigen Versionen neue Schnittstellen für weitere RTE-Systeme zu integrieren.

Wordcloud

Im Gegensatz zum vorgestellten Ansatz, den die Entwickler von Drupal gewählt haben, um Inhalte mit semantischen Informationen auszustatten (siehe Abschnitt 3.1), wurde, ähnlich dem Konzept des *One Click Annotator*, bei der Umsetzung des *Semantic Object Editor* der Fokus auf eine Fließtext-Annotation gelegt sowie auf die Anforderung, dass normale Redakteure ohne ausgeprägtes Vorwissen im Bereich des Semantic Web redaktionelle Inhalte bearbeiten können sollen. Dabei ist es besonders wichtig, auf die bisherigen Gewohnheiten und gesammelten Erfahrungen der User Rücksicht zu nehmen [24].

Um dies zu gewährleisten, wurde das semantische Interface von der Texteingabe abgekoppelt und in eine Wordcloud ausgelagert. Während im *One*

Click Annotator nicht nur die Texteingabe, sondern auch die Annotierung über ein Eingabeinterface stattfindet, das sich nahe am aktuellen Design des tinyRTE orientiert, wurde hier absichtlich auch durch einen stylistischen Umbruch die Texterstellung von der semantischen Anreicherung getrennt.

Obwohl man dadurch einerseits dem User die gewohnte Umgebung eines Editors entzieht, ist andererseits für den User durch den starken optischen Umbruch jederzeit klar, in welchem Teilinterface er sich aufhält. User sollten daher nie auf die Idee kommen, in der semantischen Ansicht Textelemente anpassen zu können. Bei einem Interface, in dem die Texteingabe als auch die semantische Annotation zur gleichen Zeit über ein zentrales Eingabefenster möglich wäre, würde die Gefahr bestehen, dass User unabsichtlich Teile der bereits integrierten Annotationen überschreiben oder semantische Platzhalter komplett oder teilweise löschen – dies könnte im weiteren Verlauf zu großen Userbility- als auch zu Darstellungsproblemen führen.

7.2.3 Semantische Anreicherung

Objektzuweisung

Die Objektzuweisung findet im Zusammenspiel mit der *Wordcloud* statt. Damit unterscheidet sich der Ansatz gravierend von Drupal, da hier Objekte erst nach dem Schreiben zugewiesen werden. Beim *One Click Annotator* findet die Zuweisung hingegen in einem RTE-inspiriertem Interface statt, um mit den gesammelten Erfahrungen der User mit *WYSIWY*-Editoren zu arbeiten [16].

Die Interaktion mit den Textobjekten in der *Wordcloud* entspricht hingegen eher dem Ansatz von sogenannten Tagclouds, die sich primär mit der gesammelten Darstellung von diversen Begriffen befassen [23]. Die grundlegende Erfahrung der User mit Tagclouds lässt sich natürlich nicht mit denen mit *WYSIWY*-Editoren gleichsetzen – sie sollte aber grundlegend soweit vorhanden sein, um sich in der Wordcloud zurechtzufinden und einzelne Objekte auswählen zu können.

Eine sogenannte *Blacklist* für Wörter und Begriffe zu erstellen, die in der *Wordcloud* nicht mehr auftauchen sollten, klingt zwar im ersten Moment verlockend – immerhin würde die Größe der *Cloud* damit um einiges reduzierbarer werden und damit eine übersichtlichere Darstellung zur Folge haben. Praktisch gesehen wäre die Integration einer Ausschlussliste hier allerdings nicht zielführend, da sich nicht ausschließen lässt, dass Wörter, die für ein Objekt nicht relevant sind, niemals für andere Objekte benötigt werden würden. Wenn der Redakteur zum Beispiel das Wort „*und*“ für unwichtig erachtet und sperrt, könnte er künftig den Firmennamen „*Muster und Partner*“ nicht mehr als Objekt auswählen, da in Folge der Sperrung nur noch „*Muster*“ und „*Partner*“ zur Auswahl stünden. Anbieten würde sich der Einsatz einer *Blacklist* hingegen bei der Objektempfehlung.

Klassen und Attribute

Das Anlegen und Zuweisen von Klassen ist in einzelne Schritte aufgeteilt und entspricht grob den Abläufen, die User in Drupal vernehmen müssen. Darin wird allerdings der Umweg über ein sogenanntes Mapping gewählt - der User legt also zuerst über Strings die gewünschten Elemente an und verbindet diese dann über ein Mapping mit ihren RDF-Pendants. Dieser Schritt findet im *Semantic Object Editor* nicht statt. Der User bekommt über die von ihm eingegebenen Strings Vorschläge von korrespondierenden Elementen aus der Datenbank geliefert und hat somit die Möglichkeit, existierende Komponenten zu wählen oder neue anzulegen, falls keine passenden gefunden werden konnten.

Der Umgang mit Klassen und Attributen ist für User sicherlich der komplizierteste Schritt, da es ein bestimmtes Grundverständnis des Semantic Web voraussetzt. Hier wäre es hilfreich, wenn unerfahrene User mit Hilfetexten durch die einzelnen Schritte geführt werden, um sicherstellen zu können, dass sie die bestimmten Bedeutungen von Objekten, Klassen und Attributen auch verstehen.

Da bei der Entwicklung jedoch darauf geachtet wurde, Redakteure mit unnötigen semantischen Funktionalitäten nicht zu überfordern und erstmal einen Editor für den Einstieg in die semantische Welt zu bieten, dürften sich User damit schnell zurechtfinden.

Probleme könnten hingegen erfahrene User haben, die mehrstufige onthologische Verbindungen zwischen Elementen erstellen möchten – dies ist mit dem aktuellen Systen, auch aufgrund der Fokussierung auf unerfahrene User, noch nicht möglich. Es wäre allerdings denkbar, dem Redakteur in künftigen Versionen die Auswahl zwischen einem *Basic*- und einem *Advanced*-Modus anzubieten, um hier die Zielgruppe der Extension auszubauen.

Objektempfehlungen

Ähnlich dem *One Click Annotator* bietet das System dem Redakteur Objektvorschläge an, die in der *Wordcloud* speziell hervorgehoben werden. Zur Zeit setzt der Editor bei der Objektanalyse dabei hauptsächlich auf syntaktische Übereinstimmungen der gelieferten Texte mit den bereits existierenden Objekten in der vom User angelegten Wissensdatenbank.

Da man davon ausgehen kann, dass sich Redakteure meist mit einem eingegrenzten Themenfeld beschäftigen und damit relativ schnell eine Datenbank erarbeitet haben, die passende Vorschläge liefert, ist der Rückschluss, dass mit der Zeit der benötigte Arbeitsaufwand stark sinkt, sicherlich korrekt und nachvollziehbar.

Andererseits bietet gerade die Objektempfehlung durch die Integration von tiefgreifenderen *Natural Language Processing*-Algorithmen oder semantischen APIs noch großen Raum zur Verbesserung (siehe Abschnitt 7.3.1).

7.3 Erweiterungsansätze

7.3.1 Tagging-Systeme

Faviki[1], *Zemanta*[2] oder *OpenCalais*[3] sind sogenannte Tagging-Systeme beziehungsweise Semantic APIs. Diese analysieren einen eingeschickten Text und liefern under anderem dazu passende Keywords zurück.

Der Vorteil der Integration solcher Systeme liegt eigentlich auf der Hand. So können komplexe *Natural Language Processing*-Algorithmen auf diese Systeme ausgelagert werden, die sich genau darauf spezialisiert haben. Es wäre sicherlich möglich, bestimmte Algorithmen auch im System selbst zu integrieren; es muss dann allerdings auch die Frage gestellt werden, wie sich diese Ergebnisse im Vergleich zu den APIs schlagen. Vorausgesetzt man kann selbst kein Expertenteam aufstellen, dass sich auf die Entwicklung und Verbesserung dieses kleinen Teilaspektes des Projekts konzentriert und die eigenen Ressourcen nicht endlos sind, ist es sinnvoller, die bereits vorhandenen Angebote zu nutzen.

Interessant wären die gelieferten Keywords speziell für die Objektempfehlungen als auch die Darstellung der Wordcloud.

Einfluss auf die Objektempfehlung

Zur Zeit kann das System nur Objekte vorschlagen, die bereits in der Datenbank angelegt wurden – durch die Unterstützung der APIs wäre es allerdings möglich, wichtige Elemente auch schon vorab zu erkennen und diese gesondert hervorzuheben. Der Vorschlag müsste sich allerdings optisch als auch technisch von einem existierenden Objekt unterscheiden. Bei Objektvorschlägen, die schon komplett annotiert und angelegt wurden, muss der User dieses nur noch als Objekt bestätigten. Bei Keywordempfehlungen muss der User allerdings das Objekt als solches noch anlegen – es gilt daher auch, passende Klassen und Attribute einzutragen.

Ideal wäre hier natürlich, wenn die APIs schon komplette URIs zu passenden Objekten und Klassen liefern würden, dann würde jede weitere Arbeit entfallen, da auf externe Objekte verlinkt werden kann. Dies würde auch dem *Linked-Data*-Prinzip entsprechen und die Vernetzung des Semantic Web voran treiben. Unter anderem bietet *Open Calais* auch RDF Feedback an, wobei die gelieferten Keywords nur als Tags annotiert sind und damit keinen Mehrwert im Sinne der Linked-Data-Cloud liefern[4].

[1] http://www.faviki.com
[2] http://www.zemanta.com
[3] http://www.opencalais.com
[4] Open Calais bietet über den *Viewer* eine Vorschau der Informationen der gelieferten Antwort an: http://viewer.opencalais.com/

Einfluss auf die Wordcloud-Darstellung

Da die gelieferten Keywords allerdings die Möglichkeit bieten, mit hoher Wahrscheinlichkeit die relevantesten Informationen aus dem Text zu filtern, würde es sich auch anbieten, diese Information in der Darstellung der *Wordcloud* einzubauen. In *Tagclouds* ist es üblich, die Terme nach Relevanz und Beliebtheit anzuordnen und auch die Darstellungsgröße der Begriffe darauf anzupassen.

Es wäre zwar unsinnig, die Begriffe in der *Wordcloud* nach der Relevanz anzuordnen, da man Gefahr läuft, den User damit mehr zu verwirren als ihn zu unterstützen. Der Ansatz, die relevanteren Begriffe durch eine eigene Formatierung hervorzuheben, könnte dem Redakteur hingegen bei der Annotierung des Textes sehr behilflich sein und damit die Userbility stark erhöhen.

7.3.2 Integration von externen Daten

Zur Zeit erlaubt es das System nicht, mit externen Daten zu arbeiten. Obwohl man mit der Integration von externem Vokabular zumindest im Ansatz den Weg in die Linked-Data-Cloud öffnet, beschränkt sich die Interaktion über den *Semantic Object Editor* damit eher auf den passiven Weg der Datenbereitstellung. Obwohl dies von Anfang an so gedacht war und der Fokus auf der Entwicklung eines Annotationsinterfaces lag, wäre es für künftige Entwicklungen sicherlich von Vorteil, wenn die Redakteure auch in der Lage wären, aktiv von der Linked-Data-Cloud zu profitieren und damit die eigene Webseite mit externen Inhalten anzureichern.

7.3.3 Pingback

Der in Kapitel 3 vorgestellte Ansatz des *Semantic Pingback* weist interessante Ansätze auf, hat allerdings das Problem, dass es erst erfolgreich funktionieren kann, wenn es in genug Systemen auch integriert wird. Es würde sich daher anbieten, eine Pingback-Schnittstelle in das System zu integrieren.

7.4 Offene Probleme

Obwohl Semantic Web als Konzept schon lange existiert, wurden erst im letzten Jahrzehnt große Entwicklungsschritte gemacht, die es vom konzeptionellen Ansatz näher an den wirklichen Webuser gebracht haben. Doch diese Evolution ist noch lange nicht abgeschlossen und daher gibt es auch noch viele grundsätzliche Probleme, die angesprochen und für die künftig von der Semantic Web-Community neue Lösungsansätze gefunden werden müssen.

7.4.1 Usermotivation

Eines der größten Probleme des Semantic Web ist immer noch die generelle Usermotivation. Nachdem man das Konzept des Semantic Web vorgestellt hat, kommt meist als Erstes die Frage nach dem „Warum?“ und „Rechnet sich dieser enorme zeitlich Aufwand denn überhaupt für mich als Webseitenbetreiber?“

Konzeptionelle Vorteile

Wie in den Grundlagen (siehe Kapitel 2) zuvor schon besprochen wurde, liegen die konzeptionellen Vorteile für User eigentlich auf der Hand: bessere Userbility-Werte für die eigene Seite durch mehr Informationsgehalt und bessere Navigation; eine deutlich gesteigerte Userbindung durch die Möglichkeit, in den Suchergebnissen der User aufzuscheinen, die auch wirklich nach der eigenen Produktpalette gesucht haben; bessere Auszeichnungen für Suchmaschinen und dadurch bessere Platzierungen; die gesteigerte Bekanntheit der eigenen Produkte durch die Veröffentlichung in der Linked-Data-Cloud, um von externen Applikationen effektiv nutz- und referenzierbar zu sein.

Wie schon beschrieben, sind diese Vorteile zur Zeit aber eher noch konzeptionell. Das bedeutet, dass zwar durch die semantische Annotierung und Veröffentlichung der Daten genau die zuvor beschriebenen Effekte auftreten können – es allerdings noch an laufenden Applikationen mangelt, die mit den erstellten Daten arbeiten und damit die möglichen Resultate überhaupt liefern können.

Passive Erfahrung

Im Semantic Web liegt der Erfolg nur zu einem bestimmten Grad am Anwender selbst. Dieser findet sich selbst schnell in der passiven Rolle des *Veröffentlichers* wieder, der zwar sein Bestes geben kann, um die vorhandenen Daten mit semantischen Informationen anzureichern und externen Quellen zur Verfügung zu stellen – wie diese daraufhin mit den Daten jedoch umgehen, liegt zum größten Teil außerhalb des Einflussbereichs des Anwenders.

Ein Problem, dessen sich die Entwickler von *Semantic Pingpack* (siehe Abschnitt 3.4.2) schon teilweise angenommen, jedoch noch nicht zufriedenstellend gelöst haben, ist das fehlende Userfeedback [32]. Da Daten aus externen Quellen ohne jegliche Verweise auf die Herkunftsquellen eingebunden werden können, ohne dass die Anbieter überhaupt wissen, wo sich die veröffentlichten Informationen überhaupt befinden und zu welchem Zweck sie verwendet werden, bleibt der User auch hier wieder in einer passiven Rolle ohne Möglichkeit, seine Daten im Internet zu verfolgen. Wie bereits angesprochen bietet der Ansatz des *Semantic Pingpack* dafür zwar eine Lösung; da dieses Modul allerdings freiwillig von Entwicklern in Applikationen eingebaut werden muss, ohne zu wissen, wie hoch die Wahrscheinlichkeit ist, dass

diese auch von anderen Anbietern gemacht wird, bleibt die Erfolgsaussicht dieses Moduls eher fraglich.

Kosten vs. Nutzen

Selbst wenn man nun die zuvor angesprochenen Vorteile als gesichert ansehen würde, steht noch immer die Frage des *Kosten/Nutzen*-Faktors im Raum. Hier liegt die Verantwortung aber eindeutig bei den Entwicklern, noch mehr dafür zu sorgen, dass semantische Applikationen so simpel und verständlich wie möglich entwickelt werden. Ein Großteil der Webuser hat von Semantic Web bisher höchstens als Schlagwort gehört – es kann also nicht erwartet werden, dass sich User und Redakteure in kürzester Zeit in komplizierte Onthologiesysteme einarbeiten.

Die Community muss daher weiter den Fokus auf die Entwicklung für Semantic Web-Applikationen für den normalen Webuser legen – es gilt, weiter auf den User zuzugehen. Mit den jüngsten Entwicklungen (siehe Kapitel 3 und Kapitel 6) wurde hier allerdings schon ein guter Anfang gemacht.

7.4.2 Verbreitung

Trotz des enormen Wachstums der Linked-Data-Cloud in den letzten Jahren, der Nutzung von Semantic Web-Portalen von öffentlichen Behörden und immer stärkerer Nutzung von Onthologieportalen speziell im Bereich der Forschung ist die prinzipielle Verbreitung des Semantic Web immer noch gering. Speziell für normale Webuser ist durch den Mangel an Applikationen der Kontakt mit Sematic-Web noch eher gering.

Accessibility

Der Einstieg in die Welt der Datenveröffentlichung ist durch den Einzug von semantischen Funktionalitäten in verbreitete Content-Management-Systeme wie Drupal bereits gegeben. Gerade in diesem Bereich ist relativ viel Engagement der Communities zu spüren und diese existierenden Systeme fungieren daher für viele User als Eintrittsschwelle in die Welt des Semantic Web.

Durch die Entwicklung von RDFa wurde es nun auch möglich, Webseiten direkt mit semantischen Annotationen anzureichern. Dieser Tatsache können sich nun natürlich auch Suchmaschinnen wie Google und Yahoo annehmen, die bereits begonnen haben, diese Informationen in ihren Suchalgorithmen zu berücksichtigen[5].

Alternativ gibt es dazu natürlich semantische Suchmaschinen wie SWSE[6] und Sindice[7], deren Webcrawler speziell mit RDF-Daten arbeiten [7].

[5] http://www.w3.org/QA/2009/05/structured_data_and_search_eng.html
[6] http://swse.deri.org/
[7] http://www.sindice.com/

Projekte wie diese befinden sich allerdings meist noch in Kinderschuhen und laufen noch nicht fehlerfrei[8].

Copyright

Interessant bleibt auch die Frage, wie mit Lizenzen und Copyright im Rahmen von Linked-Data und Semantic Web umgegangen wird. Zur Zeit werden Informationen als solche einfach für die Linked-Data-Cloud freigegeben und stehen jedem frei zur Verfügung. Da diese Daten dafür vorgesehen sind, von externen Applikationen, Personen und auch Firmen benutzt zu werden, sollte man davon ausgehen, dass dies den Veröffentlichern auch soweit bewusst ist. Ein bindender rechtlicher Rahmen existiert in diesem Sinne allerdings noch nicht, Organisationen wie *Open Data Commons*[9] [7] bieten aber Richtlinien, an denen sich die Community orientieren kann. Erwähnt werden sollte aber, dass die rechtlichen Rahmenbedingungen hier von Land zu Land variieren können.

Vertrauen in Daten

Die Fragen nach der Integrität der zu Verfügung stehenden Daten stellt ein weiteres offenes Problem dar [7]. So sind Redakteure nun in der Lage, externe Inhalte in die eigene Webseite zu integrieren. Da Websitebetreiber für den von ihnen veröffentlichten Content allerdings haften, müssen sich diese Redakteure auf den eingebundenen Inhalt vollkommen verlassen können.

Wenn man seine Inhalte von großen Portalen wie DBpedia bezieht, ist die Chance natürlich relativ hoch, dass die gelieferten Daten sich im Laufe der Zeit nicht groß ändern werden. Es ist jedoch nicht mit absoluter Sicherheit überprüfbar, ob die gelieferten Informationen faktisch richtig sind – dies obliegt jedem Redakteur selbst. Dabei kann es logischerweise problematisch werden, wenn sich diese Informationen mit der Zeit ändern, ohne dass der Redakteur davon Notiz nimmt.

Noch größere Ausmaße könnte das Problem annehmen, wenn sich der Anbieter von extern eingebundenen Inhalten nach einer gewissen Zeit als Spamanbieter herausstellt und die zuvor korrekten Informationen durch Werbung ersetzt.

Hier bedarf es weiterer Entwicklungen und Feedbackmodalitäten, die dem User mehr Kontrolle über die veröffentlichen und eingebundenen Daten geben.

[8] Bei Tests der beiden angeführten Projekte am 05.06.2011 kam es bei beiden Suchmaschinen zu Fehlern bei der Suchausführung als auch bei der Aktualität der verlinkten Elemente.

[9] http://www.opendatacommons.org/

7.4.3 Vokabular

Der Umgang mit Vokabular im Semantic Web ist ebenfalls eine zentrale Frage. Es ist, wie bereits erwähnt, *good-practice*, vordefinierte Vokabelsammlungen (siehe Abschnitt 2.5) zu nutzen und Objekte, Klassen und Attribute nur dann neu zu definieren, wenn diese noch nicht existieren.

Dabei stellt sich allerdings die Frage, wie User und Redakteure wissen sollen, was es bisher schon an vordefiniertem Vokabular gibt. Natürlich sind Sammlungen wie zum Beispiel *foaf* für die Beschreibung von Personen in Fachkreisen bekannt - dies entspricht aber nur einem kleinen Anteil der benötigten Elemente [8].

Vielmehr stellt sich die Frage, wie man das Vokabelmanagegent künftig handhaben sollte. Zur Zeit werden verschiedenste Sammlungen veröffentlicht, wobei dabei nicht einmal garantiert werden kann, dass sich die Sammlungen in bestimmten Punkten nicht überschneiden – allerdings gibt es für diese Fälle die Möglichkeiten, per *Alias*-Annotation auf die redundanten Elemente zu verweisen. Einerseits lässt sich mit den themenbezogenen Vokabularpaketen zumindest sicherstellen, dass User nicht mit einer unüberschaubaren Menge an Definitionen überrollt werden, die sie vielleicht gar nicht benötigen. Andererseits ist durch diese zerstreute Organisation viel Engagement von Anwendern nötig, damit diese an das gesuchte Vokabular überhaupt erst herankommen.

Den Usern den Umgang hierbei zu erleichtern, liegt jedoch auch an den Entwicklern. Es gilt, einfache und vor allem verständliche Wege zu finden, in denen User das benötigte Vokabular in das eigene System laden können. Dies bedeutet, dass User einerseits über existierendes Vokabular informiert werden müssen und klare Wege zum Auffinden des richtigen Vokabulars dargelegt werden sollten – die richtigen Informationen sind hier vor allem wichtig, da User sonst versucht sein werden, sich eine verwirrende Suche zu ersparen und die benötigten Elemente selbst anzulegen.

Außerdem sollten Entwickler dafür sorgen, dass in ihren Systemen ein übersichtliches *Vokabularmanagement* möglich ist. Das bedeutet, dass der Import von externen Sammlungen möglichst einfach funktionieren sollte und wenn verbreitete Sammlungen wie *foaf* und *Dublin Core* bereits vorinstalliert sind und nur noch aktiviert werden müssen, wäre dies sicherlich eine große Hilfe für Anwender. Bei vorinstalliertem, allerdings auch vom User selbst integriertem Vokabular sollte allerdings sichergestellt werden, dass regelmäßig die eingelesene Quelle nach Erweiterungen durchsucht wird. Diese kommen zwar nicht häufig vor, sollten allerdings bedacht werden, um das System aktuell zu halten.

Mit einer Vorauswahl der Entwickler ist zwar der Anfang für User leichter, wenn sie jedoch exotische Definitionen suchen, sind sie immer noch auf sich selbst gestellt und sehen sich dann mit einer zerstreuten Organisation von veröffentlichten Sammlungen konfrontiert, in der sie mit viel Glück die

gewünschten Begriffe finden – den User hier davor zu „bewahren“, die Definitionen einfach selbst anzulegen, ist von Seiten des Applikation-Entwicklers kaum zu schaffen.

Eine zentrale Verwaltungs- und Anlaufstelle würde sich hier anbieten. Damit würde einerseits für die Autoren der Sammlungen schneller ersichtlich werden, was noch fehlt und wo Überschneidungen stattfinden, andererseits würde es für normale Anwender damit möglich, sich die nötigen Informationen bei einer Anlaufstelle zu besorgen und womöglich über Suchalgorithmen genau die Definitionen und Sammlungen finden, die benötigt werden.

Ein Vokabel-Portal würde somit nicht nur den Autoren der Sammlungen, den Entwicklern der Applikationen und den normalen Webusern helfen, es würde auch dazu beitragen, repetative Vokabeldefinitionen zu vermeiden, die Verlinkung zu gleichen Elementen erleichtern und durch mehr Konsistenz die strukturelle Beschaffenheit des Semantic Web stärken.

Kapitel 8

Schlussbemerkung

8.1 Fazit

Wie sich im Laufe der Diplomarbeit im Rahmen von Thesis-Projekt und immer wieder neuen Recherchearbeiten gezeigt hat, ist Semantic Web nicht nur das neue Schlagwort für die nächste gehypte Entwicklung im Bereich des *World Wide Web*. Vielmehr handelt es sich um ein lange schlummerndes Konzept, das erst durch neue Impulse und Entwicklungen im Bereich Content-Management und Communities zusammen mit dem förmlich explosiven Wachstum der Linked-Data-Cloud in den letzten Jahren wieder erwacht ist und nun zu vielen interessanten Ansätzen und Entwicklungen geführt hat.

Die verbreitete Akzeptanz von Content-Management-Systemen und Wikis ermöglicht es im Weiteren, normale Webuser mit neuen Eingabemodulen und Modalitäten vertraut zu machen. Ansätze wie der im Rahmen dieser Arbeit entwickelte *Semantic Object Editor* oder die vorgestellten Annotationsmöglichkeiten in *Drupal* und die wegweisenden Ansätze des *One Click Annotators* befassen sich aktiv mit der noch stark verbreiteten Problematik, dass normale Webuser kaum etwas über Semantic Web wissen und sie bis vor kurzem nicht einmal Zugang zu einfach verständlichen Tools hatten.

Vor allem wurde aber auch klar, dass die großen Entwicklungen des semantischen Webs erst angefangen haben. Die größte Baustelle ist zur Zeit noch immer das *Fundament*, auf dem das Semantic Web aufgebaut werden soll. Wie in Kapitel 7 aufgezeigt, gibt es hierbei noch viele fundamentale Unklarheiten und Probleme, die es noch zu bewältigen gibt, bevor sich die Vorteile auch wirklich bezahlt machen.

8.2 Ausblick

Das Semantic Web ist hier aber auf einem sehr guten Weg und die großen geballten Entwicklungsschritte, der enorme Enthusiasmus und Einsatz in der Community und der Bedarf der Forschung als auch der Wirtschaft nach

einem strukturierterem *World Wide Web* wird dafür sorgen, dass diese Entwicklungen nicht wieder im Sand verlaufen werden.

Die Zukunft wird zeigen, wie die neuen semantischen Strukturen mit alten Technologien zusammenspielen werden. Es wird sicherlich keinen kompletten Austausch des „*alten Web*“ durch das „*neue Semantic Web*“ geben. Vielmehr wird Integration im Mittelpunkt stehen. So verband RDFa die Ausgabesprache von semantischen Informationen, RDF, mit der herkömmlichen Websprache XHTML und bekannte Suchmaschinen haben bereits angefangen, semantische Annotationen in ihre Suchalgorithmen zu integrieren.

Die „*Evolution des Web*“ ist dabei ein schönes Schlagwort, wobei sich auch hier erst zeigen muss, in welchen Gebieten semantische Annotationen wirklich sinnvoll sind und wo sie nur viel Arbeit ohne viel Nutzen mit sich bringen. Der Schlüssel liegt jedoch immer in der Verbindung zwischen Angebot und Nachfrage. Beides war bisher eher schlecht – mit den vorgestellten Entwicklungen macht das Semantic Web jedoch einen großen Schritt auf den Enduser zu und wird damit auch für Applikations-Entwickler um einiges interessanter.

Anhang A

Inhalt der CD-ROM

Format: CD-ROM, Single Layer, ISO9660-Format

A.1 PDF-Dateien

Pfad: /

Bauer_Stefan_2011.pdf	Diplomarbeit

A.2 Ressourcen

Pfad: /

Abbildungen/	In der Arbeit verwendete Abbildungen
Webquellen/	PDF-Kopien der verwendeten Webquellen

A.3 Projekt

Pfad: /Projekt/

Dokumentationen/ . . .	Dokumentationen
SourceCode/	Source-Code
T3X_SemanticObjectEditor.t3x	*Typo3* Extension

Literaturverzeichnis

[1] Adida, B., M. Birbeck, S. McCarron und I. Herman: *RDFa Core 1.1.* Website, März 2011. http://www.w3.org/TR/rdfa-core/, Kopie auf CD-ROM (docs/RDFaCore.pdf).

[2] Allsopp, J.: *Microformats: Empowering Your Markup for Web 2.0.* friends of ED, März 2007.

[3] Auer, S., S. Dietzold, J. Lehmann, S. Hellmann und D. Aumueller: *Triplify: Light-weight linked data publication from relational databases.* In: Quemada, J., G. León, Y. S. Maarek und W. Nejdl (Hrsg.): *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*, S. 621–630. ACM, 2009.

[4] Beauvais, Z.: *Best buy and the semantic web.* Nodalities Magazine, 12:16–18, Jan. 2011.

[5] Beavais, Z.: *Facebook and the open graph.* Nodalities Magazine, 10:13–14, Mai 2010.

[6] Bizer, C.: *The emerging web of linked data.* IEEE Intelligent Systems, 24(5):87–92, 2009.

[7] Bizer, C., T. Heath und T. Berners-Lee: *Linked data - the story so far.* International Journal on Semantic Web and Information Systems, 5(3):1–22, 2009.

[8] Bizer, C., A. Jentzsch und R. Cyganiak: *State of the lod cloud.* Website, März 2011. http://www4.wiwiss.fu-berlin.de/lodcloud/state/, Kopie auf CD-ROM (docs/StateOfTheLODCloud.pdf).

[9] Brickley, D. und L. Miller: *Foaf vocabulary specification 0.98.* Website, Aug. 2010. http://xmlns.com/foaf/spec/, Kopie auf CD-ROM (docs/FOAFVocabularySpecification.pdf).

[10] Clark, L. und S. Corlosquet: *Drupal: Semantic web for the masses.* Nodalities Magazine, 10:1–4, Mai 2010.

[11] Coleman, E.: *Government and data - what's the link?* Nodalities Magazine, 10:10–11, Mai 2010.

[12] Corlosquet, S., R. Cyganiak, S. Decker und A. Polleres: *Semantic web publishing with drupal.* Techn. Ber. DERI-TR-2009-04-30, DERI, Galway, Ireland, Apr. 2009.

[13] 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.

[14] Dodds, L.: *Challenges and opportunities for linked data.* Nodalities Magazine, 12:14–15, Jan. 2011.

[15] Heath, T.: *Without linked data, no semantic web.* Nodalities Magazine, 11:6, Sep. 2010.

[16] Heese, R. und M. Luczak-Roesch: *Linked data authoring for non-experts.* In: *Proceedings of the Linked Data on the Web Workshop (LDOW2009)*, Madrid, Spain, Apr. 2009. CEUR Workshop Proceedings. http://events.linkeddata.org/ldow2009/papers/ldow2009_paper4.pdf.

[17] Hepp, M.: *Goodrelations: An ontology for describing products and services offers on the web.* In: Gangemi, A. und J. Euzenat (Hrsg.): *EKAW*, Bd. 5268 d. Reihe *Lecture Notes in Computer Science*, S. 329–346. Springer, 2008.

[18] Hepp, M.: *Goodrelations language reference.* Website, Sep. 2010. http://purl.org/goodrelations/v1, Kopie auf CD-ROM (docs/GoodRelations.pdf).

[19] Hitzler, P., M. Krötzsch, S. Rudolph und Y. Sure: *Semantic Web: Grundlagen.* Springer, Berlin, 2008.

[20] Hodson, T.: *Linked data: Say what you see.* Nodalities Magazine, 12:10, Jan. 2011.

[21] Isaac, A. und E. Summers: *Skos simple knowledge organization system primer.* Website, Aug. 2009. http://www.w3.org/TR/skos-primer/, Kopie auf CD-ROM (docs/SKOS.pdf).

[22] Jin, Y., S. Decker und G. Wiederhold: *OntoWebber Model-Driven Ontology-Based Web Site Management.* In: *Semantic Web Working Symposium (SWWS)*, Stanford, California, USA, Aug. 2001.

[23] Knautz, K., S. Soubusta und W. G. Stock: *Tag clusters as information retrieval interfaces.* In: *Proceedings of the 43rd Hawaii International*

Conference on System Sciences, S. 1–10, Honolulu, Hawaii, USA, 2010. IEEE Computer Society.

[24] Luczak-Roesch, M., R. Hesse und A. Pashke: *Future content authoring.* Nodalities Magazine, 11:17–18, Sep. 2010.

[25] Manola, F., E. Miller und B. McBride: *RDF Primer.* Website, Feb. 2004. http://www.w3.org/TR/rdf-primer/, Kopie auf CD-ROM (docs/RDF_Primer.pdf).

[26] Musen, M. A.: *Bioportal: Bringing semantic technology to biomedicine.* Nodalities Magazine, 11:1–5, Sep. 2010.

[27] Noy, N. F., N. H. Shah, P. L. Whetzel, B. Dai, M. Dorf, N. Griffith, C. Jonquet, D. L. Rubin, M. A. D. Storey, C. G. Chute und M. A. Musen: *Bioportal: ontologies and integrated data resources at the click of a mouse.* Nucleic Acids Research, 37(Web-Server-Issue):170–173, 2009.

[28] Rubin, D. L., D. A. Moreira, P. P. Kanjamala und M. A. Musen: *Bioportal: A web portal to biomedical ontologies.* In: *Symbiotic Relationships between Semantic Web and Knowledge Engineering (SS-08-07)*, S. 74–77, Menlo Park, California, März 2008.

[29] Segaran, T., C. Evans und J. Taylor: *Programming the Semantic Web.* O'Reilly Media, Inc., Sebastopol, 2009.

[30] Shadbolt, N., T. Berners-Lee und W. Hall: *The semantic web revisited.* IEEE Intelligent Systems, 21:96–101, Mai 2006.

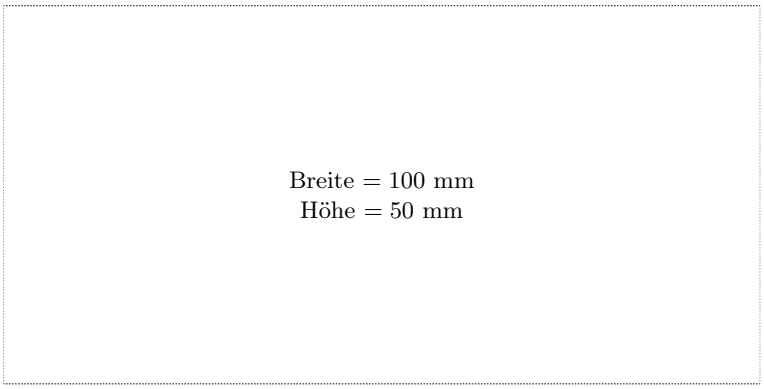
[31] Studer, R., A. Ankolekar, P. Hitzler und Y. Sure: *A semantic future for ai.* IEEE Intelligent Systems, 21:8–9, Juli 2006.

[32] Tramp, S., P. Frischmuth, T. Ermilov und S. Auer: *Semantifying pingback: Learning from the blogosphere.* Nodalities Magazine, 11:7–9, Sep. 2010.

[33] Tramp, S., P. Frischmuth, T. Ermilov und S. Auer: *Weaving a Social Data Web with Semantic Pingback.* In: Cimiano, P. und H. Pinto (Hrsg.): *Proceedings of the EKAW 2010 - Knowledge Engineering and Knowledge Management by the Masses; 11th October-15th October 2010 - Lisbon, Portugal*, Bd. 6317 d. Reihe *Lecture Notes in Artificial Intelligence (LNAI)*, S. 135–149, Berlin / Heidelberg, Okt. 2010. Springer.

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —