X3S: Eine Spezifikation zur Darstellung und interaktiven Exploration semantischer Daten

Timo Stegemann, Tim Hussein, Werner Gaulke, Jürgen Ziegler

vorname.nachname@uni-due.de

Abstract: Die zunehmenden Wandlung des Internets von einem dokumentenzentrierten hin zu einem datenorientierten Netz eröffnet für Nutzer wie Entwickler eine Fülle an Möglichkeiten. Allerdings stoßen traditionelle Ansätze des Web Engineerings bei der Fülle und Heterogenität der Daten häufig an ihre Grenzen. In diesem Beitrag wird mit X3S eine Spezifikation zur Filterung sowie zur Darstellung semantischer Inhalte des Daten-Webs vorgestellt, der das Anlegen und Wiederverwenden von Bausteinen zur Datenexploration und -visualisierung ermöglicht.

1 Einleitung

Der Einsatz semantischer Daten nimmt in den letzten Jahren stetig zu und findet sich aufgrund der hohen Ausdruckskraft in immer mehr Anwendungsbereichen. Unter anderem wird im Electronic Commerce zunehmend Gebrauch von der semantischen Repräsentation von Produktdaten [Hep06] oder Angeboten [Hep08] gemacht. Auch im Internet vollzieht sich ein Wandel: Auf der Basis von Techniken wie RDF und gefördert durch Initiativen wie Linking Open Data stehen immer mehr semantische Daten zur freien Verfügung und ermöglichen neben dem konventionellen, dokumentorientierten Web den Aufbau eines datenorientierten Webs. Hierdurch ergeben sich einerseits vielfältige Möglichkeiten der automatisierten Datenintegration und -verarbeitung, andererseits bieten semantische Daten auch dem Endanwender neue Möglichkeiten, gezielt Fakten und Zusammenhänge zu suchen. Bei der Unterstützung der Nutzer beim Umgang mit den umfangreichen und in einer Graph-Struktur hochgradig vernetzten Daten zeigen sich allerdings heute noch deutliche Probleme in Bezug auf eine einfache, verständliche Interaktion und Darstellung. Konventionelle Methoden des Web Engineering sind hierbei meist nicht hilfreich.

Um semantische Daten dem Endanwender automatisch in einer geeignet aufbereiteten Form präsentieren zu können, ist eine Methode wünschenswert, die es erlaubt, sowohl die Anfragen an das Daten-Web intuitiv zu formulieren wie auch die darzustellenden Datenelemente und deren Zusammenstellung zu spezifizieren sowie deren visuelle Repräsentation zu bestimmen. Dabei sollten die Informationen über die Daten und die Visualisierung getrennt werden („Separation of Concerns“). Zunächst ist eine zielgerichtete Auswahl und Serialisierung einzelner Datenbereiche erforderlich, um dem Nutzer ausschließlich die gewünschten Inhalte darzustellen („Filtering“). Weiterhin sollte es die Lösung ermöglichen, flexibel „Sichten“ auf einen Datenbestand oder eine Klasse von Daten zu generieren, wel-

erschienen im Tagungsband der INFORMATIK 2011
Lecture Notes in Informatics, Band P192
ISBN 978-3-88579-286-4

che dann auf strukturverwandte Daten übertragen werden können („Templating“). Die Darstellung sollte zudem flexibel gestaltet werden können, um sich an beliebige Web-Designs anzupassen („Styling“). Für Web-Entwickler sollte die Erstellung eines Stylesheets mit verhältnismäßig wenig Aufwand möglich sein („Usability (Author)“). Abschließend sollte die Benutzung für den Endbenutzer derart einfach gestaltet sein, dass er im Idealfall nicht bemerkt, dass er mit Daten des semantischen Webs umgeht („Usability (End User)“).

Im vorliegenden Beitrag werden eine Spezifikation und ein prototypischer Editor vorgestellt, die diesen Anforderungen Rechnung tragen. Zunächst werden existierende Lösungen zur Visualisierung und Exploration im Daten-Web vorgestellt und bewertet. Anschließend wird mit XSL-transformed SPARQL-results and Semantic Stylesheets (X3S) eine Spezifikation zur Definition *semantischer Stylesheets* vorgestellt. Ein solches Stylesheet beinhaltet einerseits die Möglichkeit der Datenfilterung mittels SPARQL und andererseits Darstellungsdefinitionen mittels XSL-Transformationen. Für diese Technik wurde ein Editor entwickelt, der es erlaubt, komplexe Templates für Anfragen und Ergebnisdarstellung auf einfache Weise zu formulieren und sofort einen Vorschau auf die Abfrageergebnisse zu erhalten. Abschließend vergleichen wir X3S mit den anderen vorgestellten Entwicklungen.

2 Stand der Forschung

In der Vergangenheit wurden bereits einige RDF- bzw. Semantic-Web-Browser entwickelt, mit denen es möglich war, RDF-Graphen zu explorieren und anschließend die Daten für ausgewählte Instanzen zu betrachten. Die Darstellung beschränkte sich in der Regel jedoch auf eine einfache tabellarische Auflistung aller - für diese Instanz verfügbaren - Eigenschaften (*Property*) und deren Werte (*Value*). Zwei Beispiele für solche Browser sind Tabulator[1] und Disco[2]. Eine individuelle Darstellung der Werte einer Instanz oder gar die Filterung bestimmter Werte ist mit diesen Browsern jedoch nicht möglich. Daher sollen nachfolgend Verfahren und Programme vorgestellt werden, die sich dieser spezifischen Problemstellung angenommen haben.

Die ersten beiden Verfahren, Xenon und Fresnel, beschreiben die darzustellenden Daten selbst wieder im RDF-Format. OWL-PL ist eine Adaption von XSLT für RDF-Daten und LESS bedient sich mit LeTL einer Skriptsprache, um zuvor abgerufene RDF-Daten in beliebige Textdokumente, vornehmlich HTML, zu integrieren. Zudem besitzt LESS einen einfachen Editor, welcher den Nutzer bei der Erstellung der Templates unterstützt. Im Gegensatz zu den zuvor genannten Verfahren verfügt Dido über keine eigene Beschreibungssprache, besitzt jedoch einen Editor zur Erstellung von Templates, welcher beim späteren Entwurf eines eigenen Editors als vergleichbare Anwendung herangezogen werden kann.

Xenon: Xenon ist eine von Dennis Quan vom IBM Watson Research Center und David Karger vom Massachusetts Institute of Technology entwickelte RDF-Ontologie, welche sie selbst als *„stylesheet ontology“* oder auch *„RDF stylesheet language“* bezeich-

[1] `http://www.w3.org/2005/ajar/tab`
[2] `http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/`

nen [QK05]. Ziel bei der Entwicklung war es, einem Nutzer semantische Daten auf möglichst verständliche Weise anzeigen zu können. Außerdem sollte es möglich sein, die Darstellung abhängig von den Daten und deren Kontext anzupassen.

Bei der Implementierung von Xenon orientieren sich die Entwickler stark an XSLT, ersetzten aber z.B. grundlegende Bereiche wie XPath durch SPARQL und RDQL[3], um die RDF-Daten besser verarbeiten zu können. Die Stylesheets selbst werden wieder in RDF beschrieben. Innerhalb der RDF-Daten kann HTML-Quelltext geschrieben werden, um so nach der Auswertung des Xenon-Codes durch einen entsprechenden Browser oder eine Server-Anwendung, ein strukturiertes HTML-Dokument zu erhalten.

Die schon in Quans und Kargers vorhergehender Arbeit [QK04] beschrieben Konzepte von *View* und *Lens*, werden auch in Xenon genutzt. Die Aufgabe einer Lens ist es, eine Menge von Eigenschaften einer RDF-Instanz auszuwählen. Mittels einer View wird beschrieben, wie die anzuzeigenden Daten dargestellt werden sollen. Sowohl Views wie auch Lenses können als Template definiert werden. In diesem Fall kann ihnen ein Parameter übergeben werden, welcher anschließend direkt innerhalb des Stylesheets genutzt oder auch ggf. innerhalb einer SPARQL-Anfrage ausgewertet werden kann.

Fresnel: Das RDF-Vokabular Fresnel wurde als Nachfolger von Xenon entwickelt, welches im Gegensatz zu diesem jedoch keine darstellungsspezifischen Elemente wie bspw. HTML-Tags etc. mehr enthalten soll [PBKL06]. Durch Fresnel selbst sollten nur noch die entsprechenden Daten, die für eine Visualisierung genutzt werden sollen, geliefert werden. Die eigentliche Darstellung wird von einem Browser übernommen. Unterschiedliche Browser können so auch unterschiedliche Darstellungsparadigmen nutzen. Während der eine Browser die durch Fresnel ausgewählten Daten als Tabelle anzeigt, kann ein anderer Browser diese Daten als Graph formatieren.

Da durch Fresnel nur noch die anzuzeigenden Daten, und nicht mehr wie in Xenon auch die Darstellung dieser Daten beschrieben werden sollen, wird das Konzept der *Views* nicht mehr benötigt. Die *Lenses* werden weiterhin genutzt, um die anzuzeigenden Daten zu beschreiben - die gewünschten semantischen Daten also auszuwählen und zu sortieren.

Neben den Lenses gibt es in Fresnel die sogenannten *Formats*. Durch Formats können die Daten formatiert und ggf. mit zusätzlichen statischen Daten angereichert werden. Dies können Informationen sein, die anschließend vom Browser dazu genutzt werden, die Daten mit entsprechenden CSS-Klassen zu verknüpfen. Auch kann über ein Format angegeben werden, ob ein Browser diese Daten als einen Text, einen Link, ein Bild oder sonstigen Inhalt behandeln soll. Bei einem Bild sollte dann nicht der Pfad zu diesem Bild, sondern das Bild selbst angezeigt werden und ein Link sollte sich von einem Benutzer anklicken lassen können. Lenses selbst können wieder als sogenannte *Sublenses* verschachtelt werden. Dies erhöht nicht nur die Lesbarkeit des Codes, sondern ermöglicht es auch bereits erstellte Lenses in anderen Projekten weiterzuverwenden. Anzuzeigende Eigenschaften können entweder direkt angegeben, oder, falls komplexere Anfragen nötig sein sollten, durch SPARQL oder der eigens dafür entworfene Sprache FSL (Fresnel Selector Language) abgerufen werden. FSL wurde stark durch XPath inspiriert. Sowohl mit SPARQL, wie auch mit FSL kann pro Anfrage immer nur eine Eigenschaft ausgewertet werden.

[3] `http://www.w3.org/Submission/RDQL/`

OWL-PL: OWL-PL ist eine stark an XSLT angelehnte Sprache zur Transformation von RDF/OWL in das (X)HTML-Format. Ziel bei der Entwicklung der Sprache war es, eine einfache Möglichkeit zur Darstellung von semantischen Daten zu erschaffen [Bro10]. Wie auch mit XSLT können HTML-Beschreibungen und Auswahl der Daten, beschrieben durch XML-Elemente, in einem Dokument kombiniert werden.

Neben der Sprache OWL-PL umfasst die Umsetzung zusätzlich eine Formatierungsontologie, deren Aufgabe es ist, die einzelnen RDF-Klassen oder auch Instanzen von diesen mit den, mittels OWL-PL erstellten Formatvorlagen zu verbinden. So können ganz allgemeine Voreinstellungen für Oberklassen, aber auch spezifische Darstellungsvorgaben für einzelne Objekte festgelegt werden. Um sicherzustellen, dass für jedes Objekt der Ontologie auch eine Darstellungsmöglichkeit existiert, lässt sich so auf oberster Ebene für *owl:Thing* eine Vorlage definieren, die z.B. wie Disco und Tabulator alle Eigenschaften und Werte einer Instanz in einer einfachen Tabelle anzeigt.

Damit ein Nutzer nun die mit OWL-PL formatierten semantischen Daten betrachten kann, müssen diese zuerst serverseitig ins HTML-Format umgewandelt werden. Dazu wurde die in Java geschriebene Serveranwendung VisuOWL entwickelt. Da eine RDF-Datei über die Formatierungsontologie, wie zuvor beschrieben möglicherweise mit mehreren OWL-PL-Dateien verknüpft ist, hat ein Nutzer bei VisuOWL zusätzlich die Möglichkeit, die gewünschte Darstellungsform auszuwählen.

LESS: Mit LESS wird versucht ein komplettes System[4] von der Erstellung von Templates zur Darstellung semantischer Daten, über deren Verarbeitung und Einbinden in andere Medien, bis zu deren Austausch zwischen unterschiedlichen Nutzern zu realisieren [ADD10]. Die Templates selbst werden in der eigens entwickelten deklarativen Sprache *LeTL* (LESS Template Language) beschrieben, welche, mit einigen Anpassungen, auf der Smarty Template Language[5] basiert. Dabei können die zu verarbeitenden Daten direkt aus einem RDF-Dokument ausgelesen, oder mittels SPARQL abgefragt werden. Neben den Standardfunktionen von Smarty kann LeTL daher auch RDF-Dokumente und SPARQL-Results direkt verarbeiten.

Um die Erstellung der Templates zu erleichtern, besitzt LESS einen integrierten Editor. In diesem werden in einem ersten Schritt die anzeigbaren Eigenschaften, entweder über die Angabe einer gesamten RDF-Instanz oder eine SPARQL-Anfrage definiert. Diese Eigenschaften können anschließend anhand ihrer Bezeichnung direkt innerhalb des LeTL-Codes genutzt und weiterverarbeitet werden. Die LeTL-Befehle lassen sich direkt innerhalb von HTML-Quelltext platzieren, wodurch die semantischen Daten beliebig formatiert und mittels CSS gestylt werden können. Ein Template-Autor ist allerdings nicht nur an HTML gebunden, sondern kann theoretisch auch beliebige andere Ausgaben produzieren.

Zum Schluss lässt sich das LESS-Template z.B. als IFrame oder per JavaScript in eine Webseite einbetten. Das bereits erstellte und gespeicherte Template wird mittels einer REST-Schnittstelle auf dem LESS-Server aufgerufen. Dieser generiert daraus eine Ausgabe, welche an die aufrufende Webseite zurückgeschickt wird, in welche diese Ausgabe letztendlich eingebettet werden soll.

[4] http://less.aksw.org/browse
[5] http://www.smarty.net/

Dido: Dido (*D*ata-*I*nteractive *DO*cument)[6] stellt ein aktives Dokument dar, das nicht nur die anzuzeigenden Daten enthält, sondern auch einen Editor, um diese Daten bearbeiten zu können, neue Daten hinzuzufügen oder zu löschen [KOL09]. Auch die Darstellung dieser Daten kann innerhalb des Dokuments angepasst werden.

Das Dokument selbst ist eine HTML-Datei, die sowohl die Daten wie auch den Editor enthält. Der Editor wurde in JavaScript geschrieben. Im Gegensatz zu den anderen bisher vorgestellten Verfahren arbeitet Dido nicht mit semantischen Daten im RDF-Format, sondern nutzt eine einfache Liste von Daten, bestehend aus Eigenschaften und ihren Werten, welche im Dokument im JSON-Format[7] gespeichert werden. Entsprechend besitzen die Daten auch keine allgemeingültigen URIs oder ähnliche Eigenschaften semantischer Daten.

Allerdings baut Dido auf ähnlichen Konzepten wie Xenon und Fresnel auf, bzw. setzt diese um. So werden in Dido ebenfalls die Konzepte von *Lenses* und *Views* genutzt, um Daten auszuwählen und zu formatieren. Über den Lens-Editor, können die anzuzeigenden Daten ausgewählt werden, wie auch gleichzeitig in einem HTML-Editor formatiert werden.

Mittels des View-Editors, kann bestimmen werden, auf welche Weise die Daten angezeigt werden sollen. Dazu gehören die gewöhnlichen Darstellungsformen wie die Listen- und Tabellenansicht, aber auch die Anzeige in Verbindung mit einem Zeitstrahl oder in Zusammenarbeit mit Google Maps. Diese Ansichten können auch kombiniert werden[8].

Bewertung

Xenon, OWL-PL und LeTL sind eigene Sprachen zur Beschreibung semantischer Stylesheets. Fresnel selbst beschreibt nur die zu visualisierenden Daten, aber nicht genau, wie sie darzustellen sind, wodurch Fresnel für das gewünschte Ziel nicht direkt genutzt werden kann. Ebenso wenig lässt sich Didos Editor nutzen, da dieser nicht mit Daten im RDF-Format arbeitet.

Von den drei verbleibenden Sprachen bietet LeTL die einfachste Verwendung, hat aber, wie die anderen Sprachen auch, den Nachteil, dass wenn einzelne RDF-Instanzen verschachtelt dargestellt werden sollen, auch die Stylesheets ineinander verschachtelt werden müssen. Ansonsten werden, durch die Struktur des SPARQL-Results bedingt, redundante Einträge produziert. Um die Verschachtelung von RDF-Instanz direkt mit LeTL zu ermöglichen, müsste die Sprache zuerst um eine Navigationsmöglichkeit im Result-Tree, vergleichbar zu XPath, erweitert werden.

Durch das Fehlen intuitiver Editoren zum Erstellen semantische Stylesheets, sind die vorgestellten Beschreibungssprachen für Laien kaum zu nutzen. Xenon, Fresnel und OWL-PL können so nur von Experten im Bereich semantischer Daten genutzt werden, da sowohl die darzustellenden Daten, wie auch deren Struktur für den Stylesheet-Autor bekannt sein

[6] `http://projects.csail.mit.edu/exhibit/Dido/`
[7] `http://json.org/`
[8] Eine kombinierte Ansicht von Zeitstrahl und GoogleMaps findet sich unter
`http://simile-widgets.org/exhibit/examples/presidents/presidents.html`

und verstanden werden müssen. LESS kann, wenn keine SPARQL-Anfragen erstellt werden müssen, sondern nur einfache RDF-Dokumente visualisiert werden sollen, durch die Auflistung der verfügbaren Eigenschaften innerhalb des Editors auch von einem Webentwickler genutzt werden. Nur Dido besitzt einen verhältnismäßig einfachen WYSIWYG-Editor. Frei verfügbare Implementierungen von Xenon, OWL-PL oder LeTL, die für einen eigenen Editor genutzt werden könnten, sind nicht bekannt.

Anforderungen an ein Verfahren zu Erstellung von Semantic Stylesheets

Anhand der zuvor vorgestellten verwandten Arbeiten und grundlegenden Erfahrungen mit semantischen Daten sowie dem Ziel, das Verfahren im E-Commerce-Bereich einzusetzen, lassen sich einige Anforderungen an ein Konzept zur Erstellung von Semantic Stylesheets aufstellen. Diese Anforderungen sollen im Folgenden kurz beschrieben werden, wobei sie sich in zwei Gruppen einteilen lassen: zum einen in Anforderungen, welche die Gebrauchstauglichkeit bzw. die grundlegenden Funktionen eines Verfahrens beschreiben, die erfüllt sein müssen, um das Ziel zu erreichen und zum anderen in Anforderungen, die die Benutzerfreundlichkeit des Verfahrens betreffen.

Soll das Verfahren im Bereich des elektronischen Handels eingesetzt werden, ist davon auszugehen, dass mit großen Datenbeständen, die zudem ständigen Erweiterungen und Anpassungen unterworfen sind, zu arbeiten ist. Eine Trennung von Daten und Darstellungsinformationen ist daher erforderlich („Separation of Concerns“). SPARQL hat sich als die Standardsprache für die Abfrage semantischer Daten herauskristallisiert. Ein Verfahren zur Darstellung semantischer Daten sollte dementsprechend in der Lage sein, diese Daten über das SPARQL-Protokoll abzufragen und zu verarbeiten („SPARQL-Endpoint Support“). Ebenfalls ließ sich erkennen, dass es nötig sein kann, unerwünschte Werte von der Darstellung auszuschließen. Einige semantische Datenbanken halten ihre Daten in verschiedenen Sprachversionen vor; eine Anzeige der gleichen Informationen in verschiedenen Sprachen ist in der Regel jedoch nicht gewünscht. Daten sollte sich demnach sowohl anhand ihrer Werte wie auch anhand von Metadaten, wie der Sprache herausfiltern lassen („Filtering“). Für das Verständnis der Informationen ist es förderlich, wenn sich die puren Daten in eine visuell ansprechende Form aufbereiten lassen. Dies betrifft neben dem Darstellungsformat auch die Formatierung von Form, Farbe und Größe der dargestellten Informationen („Styling“).

Eine große Stärke semantischen Daten liegt darin, dass sich unterschiedliche Klassen und ihre Instanzen leicht miteinander verknüpfen lassen. Durch Object-Properties werden ganze Instanzen zu Eigenschaften eines anderen Objekts. Da die Verknüpfung selbst aber nur einen geringen Informationswert besitzt, müssen auch die Eigenschaften eines Object-Propertys in einen Stylesheet einfließen können („Nested Object-Properties“).

Nachdem zuvor funktionale Anforderungen an ein Verfahren zur Darstellung semantischer Daten beschrieben wurden, soll nun auf eine Reihe von Anforderungen eingegangen werden, welche die Benutzerfreundlichkeit dieses Verfahrens betreffen. Während die zuvor genannten Anforderungen mindestens erfüllt sein müssen, um das gewünschte Ziel der

Erstellung eines Semantic Stylesheets zu erreichen, wird durch die nachfolgenden Anforderungen beabsichtigt, den späteren Nutzern, seinen es Software-Entwickler, Stylesheet-Autoren oder Endnutzer, die Nutzung des Verfahrens möglichst zu erleichtern.

Die zuvor vorgestellten Verfahren setzen zum Teil auf eigene Abfrage- und Beschreibungssprachen, wodurch ihre Nutzung deutlich erschwert wird. Um eine einfache Implementierung des Verfahrens zu gewährleisten, sollte das Verfahren daher in möglichst vielen Bereichen auf bereits etablierte Techniken zurückgreifen, die aufeinander aufbauend die Darstellung des Semantic Stylesheets übernehmen („Usability (Developer)“). Die Verwendung etablierter Techniken hat zudem den Vorteil, dass Stylesheet-Autoren mit diesen bereits vertraut sein können und sie so nicht in eine neue Technik einarbeiten müssen („Usability (Author)“). Für den Endnutzer sollte die Betrachtung eines Stylesheets keinen weiteren Aufwand erfordern. Im Idealfall sollte sich das Stylesheet daher in einem einfachen Webbrowser betrachten lassen („Usability (End User)“). Die Stylesheets sollten sich nicht nur für eine einzelne Instanz des Datenbestandes nutzen lassen, sondern sich auch für andere Instanzen, die zumindest der gleichen RDF-Klasse zugehörig sind, wiederverwenden lassen („Templating“).

3 X3S Spezifikation

Im folgenden Abschnitt wird mit X3S eine Spezifikation zur Beschreibung semantischer Stylesheets vorgestellt. X3S ist im Gegensatz zu den in Abschnitt 2 vorgestellten Umsetzungen keine eigene Sprache, sondern definiert überwiegend bereits bestehende Techniken in einem Workflow. Ziel des Workflows ist es, semantische Daten in ein für Menschen lesbares und optisch ansprechendes HTML-Dokument zu überführen. HTML bietet sich als Zielformat an, da es nahezu beliebig gestaltbar ist und auf allen Plattformen mit Webbrowser dargestellt werden kann. Wird der Workflow serverseitig verarbeitet, entfällt zudem die Notwendigkeit zusätzliche Plugins auf dem Client auszuführen.

Der Workflow umfasst die Verarbeitung semantischer Daten bis zur Darstellung in den vier Schritten *Datenabfrage*, *Datenumstrukturierung*, *Transformation* und *Styling*. Die meisten dieser Einzelschritte setzen auf bereits etablierte Technologien, welche durch X3S gekapselt werden. Abbildung 1 gibt einen Überblick über die einzelnen Komponenten, die im Folgenden näher betrachtet werden.

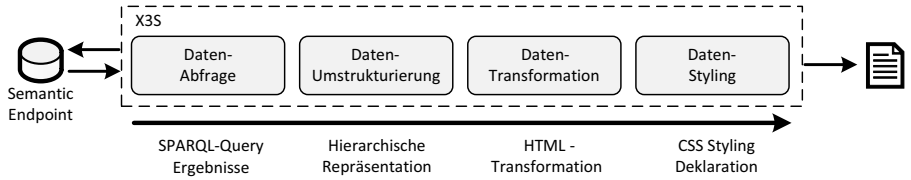

Abbildung 1: Bestandteile des X3S Formates. Die Arbeitsschritte bauen aufeinander auf, um die semantischen Daten einer beliebigen Datenquelle für die visuelle Präsentation in einem Webbrowser aufzubereiten.

1. Datenabfrage: In diesem Schritt werden die gewünschten Inhalte aus einer semantischen Datenquelle abgerufen. Die Abfrage der Daten erfolgt mit Hilfe des SPARQL-Standards[9]. So können alle semantischen Datenquellen verwendet werden, welche einen SPARQL-Endpoint zur Verfügung stellen. Durch die Etablierung von SPARQL stehen somit viele verschiedene Datenquellen wie zum Beispiel DBpedia[10] zur Verfügung. Die Ergebnisse einer SPARQL-Abfrage werden in Form von XML zurückgegeben, dessen Struktur in der entsprechenden W3C Recommendation[11] definiert ist.

Die strukturierte Form des Abfrageergebnisses ermöglicht nun eine Weiterverarbeitung der zurückgelieferten Daten, welche im nächsten Schritt erfolgt.

2. Datenumstrukturierung: Ein beanstandeter Nachteil von LeTL (vgl. 2) war es, dass Objekte nicht direkt verschachtelt dargestellt werden können, da die Ergebnisse des SPARQL-Results in einer flachen Form vorliegen, was zu redundanten Angaben im Abfrageergebnis führt. Enthält ein Objekt zum Beispiel mehrere Attribute, führt das zu einem mehrfachen Vorkommen des Objektes.

Damit die Ergebnisse des SPARQL-Results in X3S leicht genutzt werden können, ist daher eine hierarchische Abbildung erwünscht, welche die Attribute eines Objektes unter das Objekt einordnet. Ziel in diesem Prozessschritt ist daher eine Umstrukturierung des flachen SPARQL-Results in eine Baumstruktur, welche die gewünschte Objekthierarchie widerspiegelt. Die Umstrukturierung ist der einzige Schritt innerhalb des Workflows, der nicht auf bereits bestehenden Techniken zurückgreift, sondern von einem Software-Entwickler, der X3S einsetzen möchte, selbst implementiert werden muss. Für die Datenumstrukturierung wird die Objekthierarchie zuvor mittels hierarchisch verschachtelter XML-Elemente, die die Variablenbezeichnungen der SPARQL-Abfrage als Attribut besitzen, definiert. Die einzelnen SPARQL-Results werden anschließend auf diese Struktur abgebildet und aggregiert. Redundante Ergebnisse werden in einem Element zusammengefasst und unterschiedliche Ergebnisse liegen nach der Umstrukturierung als Geschwisterknoten auf einer Ebene.

Abbildung 2 verdeutlicht diesen Vorgang an einem Beispiel. In diesem Beispiel werden von einem fiktiven SPARQL-Endpoint Informationen über Universitäten in Nordrhein-Westfalen und die Städte in denen sie liegen abgerufen. Da die Universität Duisburg-Essen gleichzeitig in den Städten Duisburg und Essen beheimatet ist, liefert die SPARQL-Abfrage zwei Result-Sets zurück, die teilweise redundante Daten enthalten. Die Informationen über die Universität selbst sind in beiden Result-Sets identisch, die Informationen über die Städte unterscheiden sich jedoch.

Eine direkte Weiterverarbeitung der Result-Sets, bspw. mittels XSLT, welches im nächsten Schritt zum Einsatz kommt, würde erfordern, dass die zueinander gehörigen Ergebnisse innerhalb der XSL-Transformation gruppiert werden. In diesem einfachen Beispiel wäre eine Gruppierung noch leicht zu realisieren, doch führt jede weitere Eigenschaft, die mehrere Werte enthält zu zusätzlichen Result-Sets, in denen alle möglichen validen Kombinationen der unterschiedlichen Eigenschaften abgebildet werden. Je mehr Eigen-

[9] http://www.w3.org/TR/rdf-sparql-query/
[10] http://dbpedia.org/
[11] http://www.w3.org/TR/rdf-sparql-XMLres/

schaften abgefragt und je tiefer verschiedene Object-Properties verschachtelt werden, um so aufwendiger wird eine direkte Gruppierung durch eine deklarative Sprache wie XSLT.

Im Zwischenschritt der Datenumstrukturierung werden daher alle Result-Sets auf eine zuvor definierte, hierarchische XML-Struktur projiziert, die der Hierarchie der SPARQL-Anfrage entspricht. Redundante Informationen werden dabei vereint und das dabei entstehende XML-Dokument kann deutlich einfacher mittels XSLT in gewünschte Zielformate wie HTML umgewandelt werden. Die weitere Verarbeitung wird im nächsten Schritt vorgenommen.

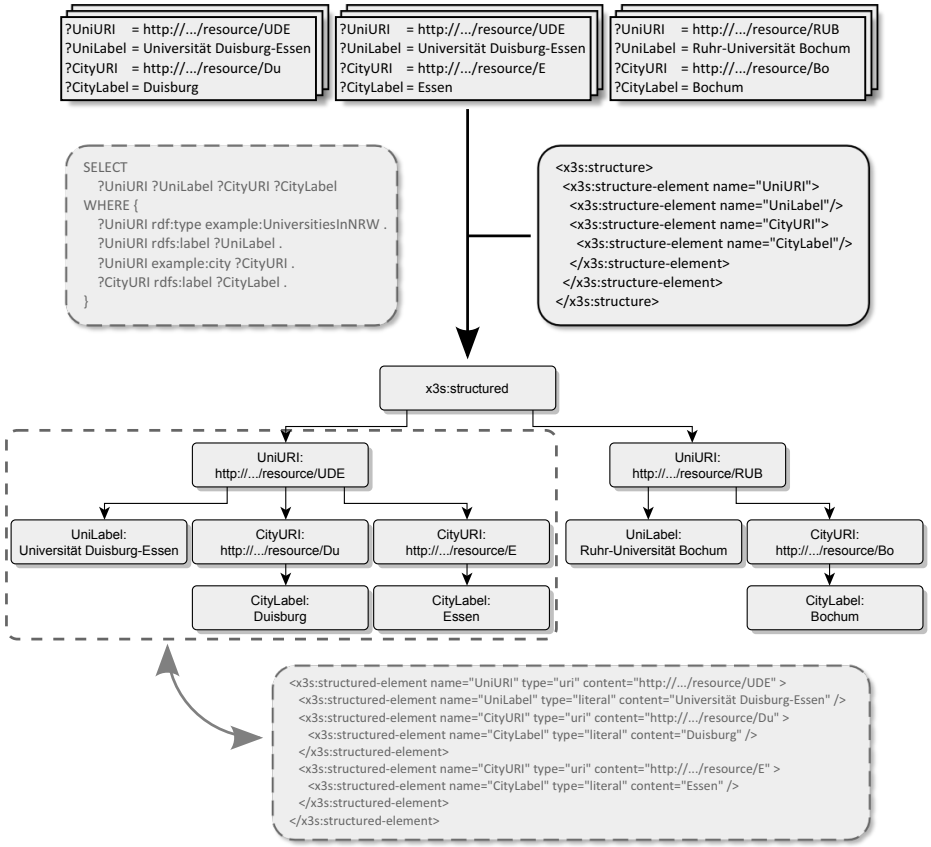

Abbildung 2: Die SPARQL-Anfrage liefert eine flache und redundanzbehaftete Liste von Ergebnissen. Diese wird anhand der Umstrukturierungsinformationen in eine hierarchische XML-Struktur überführt, in der die Ergebniswerte als Attribute der Knoten abgelegt werden.

3. Transformation: Die hierarchisch abgebildeten Ergebnisse der angefragten semantischen Daten werden in diesem Schritt in das Zielformat HTML umgewandelt. Die Umwandlung erfolgt mithilfe des etablierten XSLT 1.0 Standards[12], für welchen auf zahlreichen Plattformen Implementierungen vorhanden sind.

[12] http://www.w3.org/TR/xslt

Die Transformation überführt die Struktur in HTML-Dokumente. Neben der Generierung ganzer Dokumente können auch nur Fragmente generiert werden, welche durch Anwendungen weiter verarbeitet werden können, um diese zum Beispiel in eine Seite einzubinden. Das Designziel in diesem Schritt ist es, eine Ausgabe zu generieren die möglichst einfach dargestellt oder eingebettet werden kann.

Das generierte HTML Dokument kann an dieser Stelle bereits in gängigen Browsern betrachtet werden. Um jedoch eine optisch ansprechende Präsentation zu generieren, werden im letzten Schritt Styling Informationen hinzugefügt.

4. Styling Die verarbeiteten semantischen Daten liegen in diesem Schritt als HTML vor und können mithilfe von Cascading Style Sheets (CSS) optisch angepasst werden. Die hohe Flexibilität und Wiederverwendbarkeit von Stylesheets kommt an dieser Stelle zum Tragen. Zudem muss der Designer der Stylesheets keine Kenntnisse über die vorherigen Arbeitsschritte haben. Das designte Dokument kann nun dem Benutzer direkt angezeigt oder in einer separaten Anwendung eingebettet werden.

Zusammenfassung X3S

Die beschriebenen Arbeitsschritte von X3S lassen eine klare Trennung der Zuständigkeiten erkennen. Die Arbeitsschritte zielen darauf ab, zum einen mit möglichst einfachen und etablierten Techniken zu arbeiten und zum anderen wiederverwendbare Inhalte zu definieren. Änderung der dargestellten Elemente oder der optischen Darstellung selbst erfordert lediglich eine Änderung in der zuständigen Schicht, der Rest des Dokumentes ist wiederverwendbar. X3S kapselt somit mehrere Techniken, um Inhalte semantischer Datenquellen zu selektieren und für den Anwender darzustellen.

Zwar sind die verwendeten Techniken der einzelnen Schichten überwiegend etabliert, um jedoch normalen Anwendern die Erstellung und Verwendung von X3S-Dateien zu ermöglichen, wird im Folgenden ein interaktiver Designeditor vorgestellt. Mithilfe des Editors können X3S-Dokumente für semantische Datenquellen erstellt und betrachtet werden.

4 Ein Editor zur Erstellung von Semantic Stylesheets

Zur Erstellung semantischer Stylesheets, welche dem zuvor vorgestellten X3S-Workflow folgen, wurde ein Editor auf Basis von Adobe Flex[13] als Rich Internet Application implementiert. Damit lässt sich der Editor in jedem Webbrowser mit installiertem Flash-Player ausführen. Der Editor lässt sich für die Nutzung von Datenbeständen konfigurieren, die über eine SPARQL-Schnittstelle abrufbar sind. Dabei arbeitet er auf der Ebene von RDF-Klassen. Ein Stylesheet lässt sich anschließend also nicht nur für eine einzelne Instanz, sondern für alle Instanzen einer ausgewählten RDF-Klasse nutzen. Eine Demo-Version des Editors findet sich unter `http://interactivesystems.info/x3s`.

[13] `http://www.adobe.com/products/flex/`

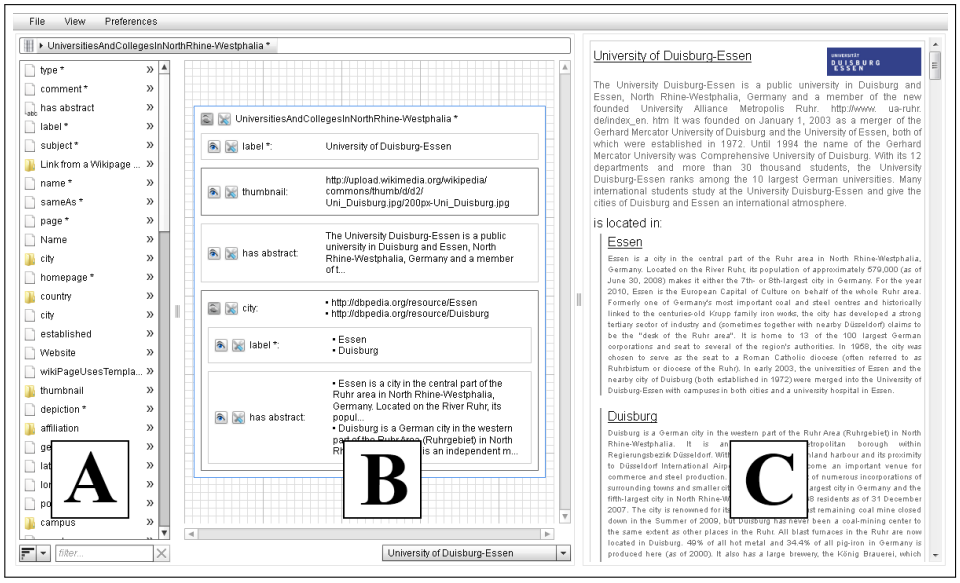

Abbildung 3: Editor zur Erstellung semantischer Stylesheets

Abb. 3 zeigt den Editor während der Erstellung eines Stylesheets. Um den Editor auch für Anwender ohne Wissen über semantische Daten nutzbar zu machen, wurde für die Erstellung eines Stylesheets, auf die textuelle Eingabe von Werten weitestgehend verzichtet.

Die Auswahl der anzuzeigenden Eigenschaften wird mittels Drag&Drop realisiert. Dabei wird dem Nutzer eine Liste (Abb. 3, A) mit allen für die RDF-Klasse verfügbaren Eigenschaften angezeigt, die er zusätzlich sortieren (alphabetisch, nach Typ, nach Relevanz[14]) und filtern kann. In Anlehnung an einen Dateiexplorer werden Datatype-Properties als Dateien und Object-Properties, als Ordner visualisiert. Datatype-Properties besitzen einen einfachen Wert, wie ein String oder eine Zahl, während Object-Properties eine Referenz auf eine weitere Instanz liefern, welche wieder weitere Eigenschaften enthalten kann.

Der Nutzer kann die Elemente aus der Liste mit der Maus, oder an einem Gerät mit Touch-Erkennung auch mit dem Finger, in den zentralen Arbeitsbereich (Abb. 3, B) ziehen. Durch die Platzierung von Object-Properties im Arbeitsbereich, werden zugehörigen Eigenschaftes des Objektes im linken Bereich aufgelistet, welche wiederum unterhalb des eingefügten Elementes im Arbeitsbereich platziert werden können. Beim Hinzufügen einer Eigenschaft (Datatype-Property) zum Arbeitsbereich, wird versucht, die passende Darstellungsform anhand einer vordefinierten Liste zu wählen. Wird zum Beispiel die Eigenschaft *depiction* oder *thumbnail* hinzugefügt, welche die URL zu einer Bilddatei enthalten sollte, wird auch später dieses Bild angezeigt statt dessen URL.

Jedes eingefügte Element kann als Filter für die generierte SPARQL-Anfrage gewählt wer-

[14] Als Relevanz wird hier die Häufigkeit des Vorkommens einer Eigenschaft unter allen Instanzen einer Klasse bezeichnet. Je mehr Instanzen einen Wert für diese Eigenschaft besitzen und in je mehr Sprachen dieser Wert existiert, als um so relevanter wird eine Eigenschaft angesehen.

den. Um die Filterung der verschiedenen Eigenschaften zu erleichtern, wird durch den Editor versucht, den Datentyp der Eigenschaft zu bestimmen, falls dieser nicht explizit durch den SPARQL-Endpoint mitgeteilt wird. Dazu werden alle Werte der Eigenschaft abgerufen und ausgewertet. Je nachdem, ob die Eigenschaft Strings, Zahlen oder bool'sche Werte enthält, werden dem Nutzer unterschiedliche Filtermöglichkeiten zur Verfügung gestellt. Abhängig vom ermittelten Datentyp werden dem Benutzer zum Beispiel Schieberegler angezeigt, mit dem er flexibel Werte definieren kann, welche bei der Datenselektion Verwendung finden.

Alle Werte lassen sich anschließend per CSS stylen. Der Editor unterstützt den Nutzer dabei, indem die gängigsten Style-Deklarationen, wie in den meisten Texteditoren auch, direkt ausgewählt werden können. So kann unter anderem Stil und Ausrichtung des Textes durch die Verwendung gängiger Texteditor-Icons bestimmt werden. Bildelemente lassen sich in Höhe und Breite anpassen. Ebenso können für alle Elemente Abstände und Rahmen definiert werden. Für Größendefinitionen können alle von CSS unterstützen Größeneinheiten verwendet. Ohne Angabe wird als Standard der Wert in Pixeln angenommen.

Die Elemente im Arbeitsbereich und die hinzugefügten Filter dienen der Generierung der SPARQL-Anfrage, welche aus dem konfigurierten Endpoint die entsprechende Ergebnismenge selektiert. Diese wird in die hierarchische Repräsentation überführt und durch den XSLT-Prozessor des Webbrowsers in HTML umgewandelt. Die durch den Benutzer konfigurierten Stylinginformationen werden in die Transformation zur Generierung von CSS einbezogen. Die Darstellung des Ergebnisses erfolgt durch einen IFrame (Abb. 3, C). Diese Verarbeitung entspricht dem vorgestellten Arbeitsschritten des X3S Konzeptes und wird bei jedem Arbeitsschritt des Benutzers ausgeführt. Der Benutzer erhält somit unmittelbar Rückmeldung, welche Auswirkung seine Aktion auf die Darstellung des Ergebnisses und der Ergebnismenge insgesamt hat. Nachdem ein Stylesheet erstellt wurde, kann es entweder als X3S-Datei exportiert, oder als statisches HTML-Dokument archiviert werden.

5 Vergleich

Nachdem sowohl die X3S-Spezifikation zur Beschreibung semantischer Stylesheets und der dazu gehörige Editor, wie auch verwandte Arbeiten vorgestellt wurden, sollen diese nun gegenübergestellt und kurz miteinander verglichen werden.

Hinsichtlich der Trennung von Daten und Layout („Separation of Concerns“) greifen, bis auf Dido, welches alle Daten zusammen mit den Styling-, Filter- und Formatierungsinformationen innerhalb eines Dokuments speichert, alle Verfahren auf separat gespeicherte Datensätze zu. Allerdings sind nur X3S und LESS dazu in der Lage diese Daten direkt aus einer semantischen Datenbank („SPARQL-Endpoint Support“) abzurufen.

Mit allen Verfahren lässt sich genau bestimmen, welche Daten angezeigt werden sollen („Filtering“). Wie diese Daten genau dargestellt werden sollen („Styling“), lässt sich mit Fresnel nicht festlegen. Die Darstellung hängt hier vom gewählten Browser ab. Die Darstellung der Daten mittels Dido lässt sich durch den eingeschränkten Editor nicht völlig beliebig bestimmten. Ein manuelles Styling direkt am Quelltext ist nicht ohne erhöhten

	X3S	Xenon	Fresnel	OWL-PL	LESS / LeTL	Dido
Separation of Concerns	●	●	●	●	●	○
SPARQL-Endpoint Support	●				●	
Filtering	●	●	●	●	●	●
Styling	●	●		●	●	○
Nested Object-Properties	●	○	○	●	○	
Usability (Developer)	●	○	○	○		
Usability (Author)	●	○	○	○	○	○
Usability (End User)	●	●	○	●	●	●
Templating	●	●	●	●	●	●

Tabelle 1: Vergleich der Verfahren. ●Trifft voll zu, ○Trifft teilweise zu, kein Eintrag: Trifft nicht zu

Aufwand möglich.

Mittels X3S können auch Werte von verschachtelten Object-Properties innerhalb eines Stylesheet problemlos anzuzeigen werden („Nested Object-Properties“), da dessen Eigenschaftswerte, genau wie die der übergeordneten Instanz, innerhalb einer SPARQL-Abfrage ermittelt und anschließend in eine hierarchische Struktur überführt werden. Für das von LESS genutzte Verfahren müssen zur Erreichung des gleichen Ergebnisses, größere Bemühung aufgenommen werden, da mehrere Stylesheets erstellen werden müssen, die sich anschließend gegenseitig aufrufen. Xenon und Fresnel gehen mit ihren Templates bzw. Sublenses ähnlich vor. Lediglich mittels OWL-PL können Object-Properties noch ähnlich leicht eingebunden werden.

Damit X3S auch von anderen Entwicklern leicht implementiert werden kann („Usability (Developer)“), haben wir versucht, uns möglichst auf bereits bestehende Techniken zu beschränken. Xenon, Fresnel und OWL-PL haben zur Beschreibung der Stylesheets eigene Sprachen entwickelt. Um diese Sprachen zu implementieren, müssen die Entwickler im Vergleich zu X3S einen deutlich hören Aufwand betreiben. Für LeTL sind keine genaueren Spezifikationen veröffentlicht worden, wodurch der Implementationsaufwand nicht bewertet werden kann. Für das geschlossene LESS-System ist eine eigene Implementation ohnehin nicht vorgesehen.

Weder für Xenon noch Fresnel oder OWL-PL existiert ein Editor, der einen Nutzer bei der Erstellung von Stylesheets unterstützt („Usability (Author)“). Die Editoren von LESS und Dido erleichtert die Arbeit zwar, sind unserer Meinung nach aber zu kompliziert, um auch von einem Laien im Bereich der semantischen Daten genutzt werden zu können. Ein Hauptaugenmerk der Arbeit lag daher auch auf der Implementation eines einfach zu nutzenden Stylesheet-Editors. Die letztendliche Betrachtung der Daten durch den Endnutzer, stellte für diesen meist kein weiteres Hindernis dar („Usability (End User)“), da die Daten entweder serverseitig ins HTML-Format transformiert wurden, oder diese Transformation direkt im Web-Browser des Nutzers geschieht. Um mittels Fresnel beschriebene Stylesheets betrachten zu können, muss der Nutzer jedoch einen eigenen Browser benutzen. Unabhängig von dem gewählten Verfahren lassen sich die Stylesheets immer für eine

Gruppe von Daten oder RDF-Klasse wiederverwenden („Templating“) und nicht nur für eine einzelne Instanz aus dieser Menge.

6 Diskussion

In diesem Beitrag wurde mit X3S eine Spezifikation zur Filterung wie Darstellung semantischer Inhalte des Daten-Webs vorgestellt, der das Anlegen und Wiederverwenden von Bausteinen zur Datenexploration und -visualisierung ermöglicht. Dabei unterstützt X3S eine Trennung von Daten und Layout; es erlaubt das beliebige Filtern von Daten wie auch frei wählbare Darstellungen mittels CSS-Direktiven. Bei der Entwicklung von X3S wurde besonderes Augenmerk auf die einfache Verwendung für Entwickler wie Endnutzer gelegt.

Die vorgestellte Technik bietet vielfältige Möglichkeiten, den Einsatz von semantischen Daten in unterschiedlichen Anwendungsbereichen zu unterstützen. Allgemein können sie die Einbindung semantischer Informationen in Webanwendungen erleichtern. Beispielsweise können vordefinierte, X3S-Templates zur Darstellung unterschiedlicher Konfigurationen von Produktinformationen bei der Produktsuche herangezogen werden. Sie eignen sich aufgrund ihrer einfachen Handhabung weiterhin zum Aufbau intuitiver Explorationsumgebungen für semantische Daten, beispielsweise von Kundenberatungssystemen auf Basis von Multitouch-Oberflächen. Zukünftige Arbeiten zur Weiterentwicklung zielen darauf ab, eine generische Verarbeitungskomponente für X3S zur Verfügung zu stellen und diese im Kontext unterschiedlicher Anwendungssysteme zu erproben. Ebenso sind empirische Untersuchungen zur Einsetzbarkeit und Gebrauchstauglichkeit für Endanwender geplant, die auf Basis dieser Technik einen erleichterten Zugang zu dem wachsenden Bestand an semantischen Daten erhalten sollen.

Danksagungen

Die in diesem Beitrag präsentierte Arbeit ist im Rahmen des Forschungsverbunds *Kontextadaptive Interaktion in kooperativen Wissensprozessen* (CONTici) entstanden. CONTici wird gefördert durch die Deutsche Forschungsgemeinschaft. Neben der Universität Duisburg-Essen sind die Universität Siegen, die RWTH Aachen sowie die Fernuniversität in Hagen an CONTici beteiligt.

Literatur

[ADD10] Sören Auer, Raphael Doehring und Sebastian Dietzold. LESS - Template-Based Syndication and Presentation of Linked Data. In Lora Aroyo, Grigoris Antoniou, Eero Hyvönen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral und Tania Tudorache, Hrsg., *ESWC (2)*, Jgg. 6089 of *Lecture Notes in Computer Science*, Seiten 211–224. Springer, 2010.

[Bro10] Matt Brophy. OWL-PL: A presentation language for displaying semantic data on the web. Diplomarbeit, Lehigh University, 2010.

[Hep06] Martin Hepp. Products and Services Ontologies: A Methodology for Deriving OWL Ontologies from Industrial Categorization Standards. *Int'l Journal on Semantic Web & Information Systems (IJSWIS)*, 2(1):72–99, 2006.

[Hep08] Martin Hepp. GoodRelations: An Ontology for Describing Products and Services Offers on the Web. In Aldo Gangemi und Jérôme Euzenat, Hrsg., *Knowledge Engineering: Practice and Patterns*, Jgg. 5268 of *Lecture Notes in Computer Science*, Seiten 329–346. Springer Berlin / Heidelberg, 2008.

[KOL09] David R. Karger, Scott Ostler und Ryan Lee. The Web Page as a WYSIWYG End-User Customizable Database-Backed Information Management Application, 2009.

[PBKL06] Emmanuel Pietriga, Christian Bizer, David R. Karger und Ryan Lee. Fresnel - A Browser-Independent Presentation Vocabulary for RDF. In *In: Proceedings of the Second International Workshop on Interaction Design and the Semantic Web*, Seiten 158–171. Springer, 2006.

[QK04] Dennis Quan und David R. Karger. How to Make a Semantic Web Browser. In *Proceedings of the 13th international conference on World Wide Web*, Seiten 255–265, 2004.

[QK05] Dennis Quan und David R. Karger. Xenon: An RDF Stylesheet Ontology. In *In WWW 2005*, 2005.

ISBN 978-3-88579-286-4