Freie Universität Berlin
Institut für Informatik

Repositorien und das Semantic Web – Repositorieninhalte als Linked Data bereitstellen

Pascal-Nicolas Becker

Diplomarbeit

Erstgutachter: Prof. Dr. Adrian Paschke
Zweitgutachter: Prof. Dr. Carsten Schulte

Eingereicht am 17. Juni 2014
Verteidigt am 02. Juli 2014

Zusammenfassung

Repositorien sind Systeme zur sicheren Speicherung und Publikation digitaler Objekte und der die Objekte beschreibenden Metadaten. Repositorien bieten ihre Inhalte hauptsächlich über Weboberflächen an, die auf die menschliche Wahrnehmung ausgerichtet sind. Die gespeicherten Daten stellen sie nicht so bereit, dass sie einfach durch Computer verarbeitet werden können oder sie verstecken sie hinter spezifischen Schnittstellen. Dabei eignen sich die in Repositorien gehaltenen Daten besonders gut dazu, als Linked Data in das Semantic Web eingebunden zu werden, da Metadaten bereits vorhanden sind und nicht erst generiert oder manuell erfasst werden müssen. Wie Inhalte von Repositorien in die Linked (Open) Data Cloud eingebunden werden können und welche Besonderheiten von Repositorien dabei beachtet werden müssen, sind die zentralen Fragestellungen dieser Arbeit.

Abstract

Repositories are systems to safely store and publish digital objects and their descriptive metadata. Repositories mainly serve their data by using web interfaces which are primarily oriented towards human consumption. They either hide their data behind non-generic interfaces or do not publish them at all in a way a computer can process easily. At the same time the data stored in repositories are particularly suited to be used in the Semantic Web as metadata are already available. They do not have to be generated or entered manually for publication as Linked Data. The main topics of this thesis are how metadata and digital objects stored in repositories can be woven into the Linked (Open) Data Cloud and which characteristics of repositories have to be considered while doing so.

DOI: http://dx.doi.org/10.14279/depositonce-5015

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die meisten Informationen im Internet sind nach wie vor auf den menschlichen Konsum ausgerichtet. Bereits 1999 schuf Tim Berners-Lee den Begriff *Semantic Web*, der für ein Netz steht in dem Informationen auch so angegeben werden, dass Computer sie verarbeiten können.[1]

2001 folgte ein viel beachteter Artikel von Berners-Lee, Hendler und Lassila, in dem sie eine Vision ausmalten, die den Nutzen für Menschen verdeutlichte, wenn Computer im Internet vorhandene Informationen in Bezug zueinander setzen und aus verknüpften Informationen Schlussfolgerungen ziehen könnten.[2] Damit zeigten sie den Bedarf auf, Informationen auch maschineninterpretierbar bereitzustellen.

Linked Data wurde seitdem zum Begriff für Daten, die in einer maschineninterpretierbaren Form Informationen bereitstellen. Grundlage dazu bilden das Resource Description Framework (RDF) als Datenmodell, verschiedene Serialisierungen für Daten in RDF und Konventionen, die sich als nützlich erwiesen haben, wenn es darum geht Informationen für das Semantic Web bereitzustellen. Die Frage, wie Informationen für das Semantic Web bereitzustellen sind, scheint auf den ersten Blick beantwortet: Es gibt etliche Publikationen, die dieser Frage nachgehen. Einen Überblick über die formalen Standards gibt zum Beispiel Hitzler u. a. 2008, für Konventionen rund um Linked Data sind Berners-Lee 2006, Bizer, Cyganiak u. a. 2007 und Bizer, Heath u. a. 2009 empfehlenswert. Bei der Erstellung neuer Informationen kann man diese Standards und Konventionen von Beginn an berücksichtigen. Geht es jedoch darum, bereits vorhandene Informationen auch als Linked

[1] Vgl. zum *Semantic Web* Berners-Lee und Fischetti 1999, S. 191: „This creates what I call a *Semantic Web* – a web of data that can be processed directly or indirectly by machines.“ *Web of Data* und *Semantic Web* werden dem Zitat entsprechend oft synonym verwendet.

[2] Berners-Lee, Hendler u. a. 2001.

Data anzubieten, muss darauf geachtet werden, wie diese Informationen bislang bereitgestellt wurden und wie sie genutzt werden. Ist die Speicherung und/oder die Bereitstellung von Daten integraler Bestandteil eines Systems, auf den andere Funktionen aufsetzen, braucht es ein dediziertes Konzept für die Einbindung solcher Systeme in das Semantic Web. Optimal ist es, wenn dabei durch Techniken des Semantic Web bestehende Probleme gelöst werden können.

Ein Beispiel dafür sind Systeme, für die sich im Umfeld von Bibliotheken der englische Begriff *repository* durchgesetzt hat, der ins Deutsche meist mit *Repositorium* übersetzt wird. Der Begriff ist jedoch nicht fest definiert und umfasst eine unscharfe Menge verschiedener Systeme.[3] Prinzipiell dienen Repositorien der sicheren Speicherung digitaler Objekte[4] und der die Objekte beschreibenden strukturierten Metadaten[5]. Durch diese Art, Daten zu speichern, auffindbar und durchsuchbar zu machen, unterscheiden sich Repositorien von anderen Systemen zur sicheren Speicherung digitaler Objekte.

Die Einbindung von Repositorien in das Semantic Web ist aus mehreren Gründen erstrebenswert. In den meisten Repositorien werden die Metadaten von Nutzern[6] manuell eingegeben und anschließend von Bibliothekaren kontrolliert und ergänzt. Daher kann in Repositorien von qualitativ höherwertigen Metadaten ausgegangen werden, als zum Beispiel im Fall von automatisch generierten Metadaten.[7] Repositorien enthalten bereits strukturierte Metadaten, die nicht erst generiert, durch aufwendige Algorithmen aus Freitexten extrahiert oder nachträglich manuell

[3] Aschenbrenner und Neuroth 2011, S. 102.

[4] Im Folgenden werden alle in einem Repositorium gespeicherten Informationen als *digitales Objekt* bezeichnet. Ein digitales Objekt kann je nach Repositorium ein Dokument, ein Forschungsdatensatz, ein Bild oder auch jeder beliebige andere Datensatz sein. Die Nutzung des Begriffs soll vermitteln, dass die Daten eines digitalen Objekts als eine Einheit angesehen werden, unabhängig davon, ob sie in einer oder mehreren Dateien gespeichert sind. Diese generische Anwendung des Begriffs sollte nicht verwechselt werden mit der Definition, die zum Beispiel Andreas Aschenbrenner und Heike Neuroth (Aschenbrenner und Neuroth 2011, S. 102) angeben und die neben den hier als digitales Objekt bezeichneten Daten auch die zugehörigen Metadaten umfasst.

[5] Mit *strukturierten Metadaten* sind Metadaten gemeint, die nicht zum Beispiel als Fließtext vorliegen, sondern in einer Form, bei der Art und Wert eines Metadatums festgelegt sind.

[6] Aus Gründen der besseren Lesbarkeit wird hier und im Folgenden lediglich die männliche oder weibliche Form verwendet. Alle Personen, unabhängig von ihrem Geschlecht, sind darin jeweils implizit eingeschlossen.

[7] Vgl. zur Qualität manuell erstellter und generierter Metadaten Hinze u. a. 2012. Die Qualität von Metadaten in Repositorien scheint bislang nicht detailliert untersucht worden zu sein. Der „2012 Census of Open Access Repositories in Germany“ (Vierkant u. a. 2012) und die Folien des Vortrags „Was ist sichtbar? Status Quo und Zukunft der Erschließung von wissenschaftlichen Inhalten in deutschen Open-Access-Repositorien“ von Maxi Kindling und Paul Vierkant auf der 12. Inetbib-Tagung 2013 (http://de.slideshare.net/paulvierkant/census-oanvortrag, abgerufen am 06.06.2014) geben erste Anhaltspunkte.

erzeugt werden müssen, sondern lediglich in eine für das Semantic Web nutzbare Form umzuwandeln sind. Des Weiteren kommen bislang spezielle Schnittstellen im Umfeld von Repositorien zum Einsatz. Daher ist es zur Nutzung der Daten eines Repositoriums erforderlich, Clients zu programmieren oder anzupassen, sowie Konkordanzen zwischen Metadatenschemata zu erstellen. Würden Metadaten und digitale Objekte im Sinne von Linked Data angeboten, das heißt als RDF und vielleicht über einen SPARQL-Endpoint, wäre ihre Einbindung und Nachnutzung deutlich einfacher.

Es gibt verschiedene Softwarelösungen für Repositorien, jedoch keine, die die gespeicherten Inhalte in das Semantic Web einbindet.[8] Eine Erweiterung bestehender Repositoriensoftware kann dafür sorgen, dass nicht nur eine Sammlung von Datensätzen für das Semantic Web bereitgestellt wird, sondern viele verschiedene Repositorienbetreiber in die Lage versetzt werden, ihre Daten als Linked Data zu publizieren.

Die Speicherung digitaler Objekte und ihrer Metadaten ist die Kernaufgabe eines Repositoriums. Oft zählt auch das Bereitstellen und der Austausch dieser Informationen dazu. Um solche Systeme, bei denen die Speicherung und Bereitstellung von Daten im Mittelpunkt stehen, in das Semantic Web einzubinden, bedarf es eines auf das System zugeschnittenen Konzepts. Im Fall von Repositorien muss es Standards und Konventionen rund um Linked Data, Eigenarten von Repositorien sowie Konventionen aus dem Repositorienumfeld berücksichtigen. Wie in Repositorien gespeicherte Metadaten und digitale Objekte als Linked Data bereitgestellt werden können und welche Besonderheiten von Repositorien dabei beachtet werden müssen, sind die zentralen Fragestellungen dieser Arbeit.

Bei der Öffnung von Repositorien in Richtung des Semantic Web ist zwischen der direkten Einbindung der digitalen Objekte und der Bereitstellung ihrer Metadaten zu differenzieren. Haben wir es zum Beispiel mit einem Objekt zu tun, das eine Reihe von Messwerten enthält, so ist es wünschenswert, die Werte für sich als Linked Data anzubieten. Handelt es sich bei dem Objekt jedoch um eine Datei, deren Inhalt als Ganzes und im Binärformat zur sinnvollen Nutzung erforderlich ist, so ist eine Beschreibung des Objekts und ein Link auf die Datei ins Semantic Web einzubringen. Diese Entscheidung ist vom jeweiligen Objekt abhängig und entsprechend aufwändig. Es kann nicht davon ausgegangen werden, dass alle Einrichtungen, die ein Repositorium betreiben, dazu bereit und in der Lage sind. Als Erfolg könnte bereits angesehen werden, wenn die Objekte von Repositorien in einem

[8] Die einzige dem Autor bekannte Repositoriensoftware, die den Export von Daten in RDF ermöglicht, ist EPrints. In Kapitel 3 werden jedoch Mängel bei der Umsetzung aufgezeigt, so dass zwar von einer Konvertierung in RDF, jedoch nicht von einer Einbindung in das Semantic Web gesprochen werden kann.

ersten Schritt im Semantic Web verlinkt und durch die vorhandenen Metadaten beschrieben würden. Entsprechend steht in dieser Arbeit die Umwandlung der Metadaten in Linked Data und die Verlinkung der digitalen Objekte in ihrer derzeitigen Form im Vordergrund.

Moderne Repositoriensoftware kann nicht nur Metadaten aufnehmen, die bei der Entwicklung berücksichtigt wurden, sondern flexibel an den Einsatzzweck angepasst werden. Beim Konzept zur Einbindung von Repositorien in die Linked Data Cloud ist dies zu berücksichtigen. Deshalb wird in dieser Arbeit nicht ein bestimmtes Metadatenschema in RDF umgesetzt, sondern ein Konzept erarbeitet, das sich auf beliebige Metadatenschemata anwenden lässt. Auf einzelne Ontologien[9] oder Vokabulare wird nur insoweit eingegangen, wie sie als Teil des erstellten Konzeptes von Bedeutung sind. In den Einrichtungen, die ein Repositorium betreiben, kann also entschieden werden, welche Ontologien und Vokabulare für die einzelnen Metadatenfelder genutzt werden. Das in dieser Arbeit erstellte Konzept ermöglicht ein entsprechend flexibles Vorgehen bei der Transformation der Metadaten in Linked Data. Das Potential von Linked Data kommt erst zum Tragen, wenn Daten verlinkt werden. Neben einer Transformation der Metadaten in RDF wird daher eine Möglichkeit erarbeitet, Metadaten in Links auf bereits als Linked Data veröffentlichte Daten umzuwandeln.

Der erste Teil der Arbeit setzt sich mit Repositorien auseinander: In Kapitel 2 wird der sonst sehr weit gefasste Begriff *Repositorium* eingegrenzt. Dies ist für die weitere Arbeit eine wichtige Grundlage, da es nach Kenntnis des Autors in der Literatur bislang keine allgemeingültige Definition des Begriffs Repositorium gibt. Anschließend wird der bisherige Austausch von Metadaten und digitalen Objekten mit und zwischen Repositorien dargestellt. Dabei werden die Probleme herausgearbeitet, die durch den Einsatz spezifischer Schnittstellen entstehen, und dargestellt, wie sich diese – durch die Einbindung von Repositorien in das Semantic Web – lösen lassen.

In Kapitel 3 wird der aktuelle Forschungsstand der im Folgenden relevanten Aspekte von Linked Data zusammengefasst und gängige Repositoriensoftware vorgestellt. Dazu werden zunächst Konventionen zur Publikation von Daten als Linked Data dargestellt und untersucht, wie sie mit Konventionen aus dem Umfeld von Repositorien in Einklang gebracht werden können. Außerdem werden besondere Eigenschaften von Repositorien und die daraus resultierenden Herausforderungen an

[9] *Ontologie* wird im Semantic Web als Begriff für Dokumente genutzt, in denen sogenanntes *terminologisches Wissen* oder *Schemawissen* bereitgestellt wird. Dazu werden Termini, semantische Relationen zwischen den Termini und Regeln spezifiziert. Über die Regeln können Integritätsbedingungen abgebildet werden oder sie dienen dazu logische Schlussfolgerungen aus Aussagen zu ziehen, in denen die spezifizierten Termini genutzt wurden.

ein Konzept zur Einbindung von Repositorien in das Semantic Web erarbeitet. Eine Übersicht der bisherigen Ansätze zur Einbindung von Repositorien ins Semantic Web schließt das Kapitel ab.

In Kapitel 4 wird ein Konzept für die Einbindung von Repositorien in das Semantic Web erarbeitet. Ziel ist die Entwicklung einer Vorgehensweise für die Umwandlung der in Repositorien gehaltenen Informationen in Linked Data unter Berücksichtigung der Eigenarten von Repositorien. In einer Erweiterung von DSpace, der weltweit am meisten eingesetzten[10] Open-Source-Software für Repositorien, wird das erarbeitete Konzept in einer Proof of Concept Implementation umgesetzt. Eine produktreife Umsetzung würde viele Repositorienbetreiber weltweit in die Lage versetzen die Inhalte ihrer Repositorien auch als Linked Data bereitzustellen.

Eine Evaluation des Konzepts in Kapitel 5 und ein Ausblick auf weiterführende Forschungsfragen in Kapitel 6 schließen die Arbeit ab.

[10] Vgl. Abschnitt 3.2.1 ab S. 37.

Kapitel 2

Grundlagen von Repositorien

Das Internet hat unseren Zugang zu Informationen und Daten aller Art maßgeblich verändert. In Bezug auf wissenschaftliche Publikationen hat sich das spätestens in der Open-Access-Bewegung niedergeschlagen. Seit inzwischen etlichen Jahren entstehen immer mehr digitale Repositorien, die wissenschaftliche Publikationen, Forschungsdaten oder andere digitale Objekte[11] speichern.

Doch was ist ein Repositorium? Welche Anforderungen sind an Repositorien zu stellen und wie unterscheidet sich ein Repositorium von einer einfachen Publikationsliste auf der Homepage eines Wissenschaftlers? Wie erfolgt der Austausch digitaler Objekte zwischen Repositorien, wie der Export digitaler Objekte aus Repositorien und was sollte noch getan werden, damit das Wissen, das in einem Repositorium gespeichert ist, einfach nachgenutzt werden kann?

Die Frage, was ein Repositorium ist, dürfte umfangreich genug sein, um ihr eine eigene Arbeit zu widmen. Repositorien werden in so vielen heterogenen Umgebungen für so viele verschiedene Aufgaben und mit so vielen unterschiedlichen Schwerpunkten realisiert, dass es schlicht unmöglich erscheint, die vielfältigen Erscheinungsformen von Repositorien vollständig zu erfassen. Entsprechend unübersichtlich ist die vorhandene Literatur. Obwohl es eine große Anzahl von Vorträgen und Artikeln gibt, in denen einzelne Repositorien vorgestellt oder einzelne Aspekte von Repositorien dargestellt werden, gibt es bis heute „keine universelle Definition oder zeitlose Standards“[12] von und über Repositorien. Stattdessen werden Repositorien in der Literatur meist in einem spezifischen Kontext, zum Beispiel als institutionelle Repositorien, oder hinsichtlich spezieller Aufgaben wie der Langzeitarchivierung dargestellt.

[11] Vgl. Fußnote 4 auf Seite 6.

[12] Aschenbrenner und Neuroth 2011, S. 102.

Da in den folgenden Kapiteln ein möglichst allgemeines Konzept für die Einbindung von Repositorien in das Semantic Web erarbeitet wird, müssen wir uns mit dem Begriff Repositorium umfassend auseinandersetzen. Dazu erarbeiten wir eine allgemeine Definition des Begriffs, die sich auf eine große Menge sehr heterogener Systeme anwenden lässt.

Abschließend wird untersucht wie Repositorien ihre Inhalte bislang bereitstellen und austauschen. Dabei werden eine Reihe von Problemen deutlich, die zu einem großen Teil durch eine Einbindung in das Semantic Web gelöst werden könnten. Die Vorteile einer Einbindung von Repositorien in das Semantic Web werden am Ende des Kapitels dargestellt.

2.1 Begriffsklärung: Repositorium

Seit dem Aufkommen der ersten Dokumentenserver[13] wurden die verschiedensten Repositorien aufgebaut. Wir wollen uns dem Begriff *Repositorium* zunächst durch die Betrachtung dreier Kriterien nähern, die oft herangezogen werden, um Repositorien gegeneinander abzugrenzen:

- Was für digitale Objekte sammeln sie?

  Die bekanntesten Vertreter von Repositorien sind Dokumentenserver, die sich meist den Austausch oder die Archivierung wissenschaftlicher Publikationen zum Ziel setzen. Inzwischen gibt es aber auch Repositorien für die verschiedensten Arten von digitalen Objekten, zum Beispiel bestimmte Bilddatenbanken, Repositorien für Audiodateien, Forschungsdatenrepositorien und vieles mehr.

- Welche Regeln gibt es zur Aufnahme neuer Objekte in das Repositorium?

  Meist wird zwischen institutionellen und fachspezifischen Repositorien unterschieden. Institutionelle Repositorien verlangen in der Regel einen Bezug der aufzunehmenden Objekte zur zugehörigen Institution. Zum Beispiel könnte das Projekt, in dessen Rahmen die zu speichernden Objekte entstehen, an der Institution angesiedelt oder eine Mitarbeiterin oder ein Mitarbeiter an der Erstellung der Objekte beteiligt sein. Fachspezifische Repositorien setzen einen inhaltlichen Bezug voraus, das heißt, die Objekte, die in ein fachspezifi-

[13] Als eines der ersten Repositorien kann sicherlich arXiv angesehen werden, siehe dazu Ginsparg 1994, worin der Aufbau des Repositoriums beschrieben wird, das später in arXiv umbenannt wurde.

sches Repositorium aufgenommen werden sollen, müssen einen inhaltlichen Bezug zur entsprechenden Disziplin aufweisen.

- Welche Regeln gibt es für den Zugriff auf Objekte des Repositoriums?

  Die Regeln für den Zugriff auf Objekte eines Repositoriums sind vielfältig. Open-Access-Repositorien setzen die freie Verfügbarkeit der digitalen Objekte voraus. Es gibt aber etliche Repositorien, die Zugriff auf die gespeicherten Objekte nur zum Teil, unter speziellen Bedingungen oder gar nicht gewähren. Übliche Einschränkungen sind, dass Zugriffe nur aus der betreibenden Institution heraus vorgesehen sind oder Gebühren für den Zugriff auf Objekte erhoben werden. Die Speicherung von Objekten, auf die kein Zugriff gewährt wird, mag im ersten Moment befremdlich sein. Hintergrund kann aber zum Beispiel die Langzeitarchivierung in Verbindung mit langen Sperrfristen sein.

Aus diesen Kriterien zur Abgrenzung von Repositorien gegeneinander wird bereits deutlich, wie schwierig es ist eine Definition für den Begriff *Repositorium* zu finden. Mit den verschiedenen Umgebungen, in denen Repositorien zu finden sind, und den unterschiedlichen Funktionsanforderungen an die verschiedensten Repositorien haben Aschenbrenner und Neuroth begründet, dass es keine Definition des Begriffs gibt.[14] Dennoch scheint es einen Konsens darüber zu geben was ein Repositorium ist. Denn der Begriff wird bis heute mindestens im Umfeld von Bibliotheken als fester Terminus genutzt. Eine Lösung dieses Problems kann darin liegen, von konkreten Repositorien zu abstrahieren und zwischen Zielen vieler Repositorien und speziellen Zielen einzelner Repositorien zu differenzieren. In diesem Sinn wird der Begriff im Folgenden eingegrenzt. Betrachten wir dazu die drei Grundfunktionen eines Repositoriums. Ein Repositorium

- sammelt digitale Objekte,
- nimmt diese Objekte strukturiert auf und
- speichert die aufgenommenen Objekte sicher.

Ein digitales Objekt kann zum Beispiel ein digital vorliegender Text, ein Bild, ein Datensatz aus einer Messreihe oder eine Tabelle mit Messwerten sein. Letztlich geht es um eine Information, die von digitalen Daten repräsentiert wird. Diese Daten können in einem Repositorium gespeichert und von einem Menschen – in der Regel unter Zuhilfenahme eines Computers – interpretiert werden, wodurch die ursprüngliche Information wieder entsteht.

[14] Vgl. Aschenbrenner und Neuroth 2011, S. 102.

Das strukturierte Aufnehmen eines digitalen Objekts zielt darauf ab, durch eine gewisse Ordnung das Objekt leicht finden, beziehungsweise ein Objekt beschreiben zu können. Die Struktur gibt vor, wie das Objekt zu beschreiben ist und welche Angaben zum Objekt erforderlich sind, damit das Repositorium das Objekt verwalten kann. Bei der strukturierten Beschreibung eines Objekts handelt es sich um Metadaten, die im Repositorium zusammen mit dem Objekt selbst abgelegt werden. Dabei ist wichtig, dass die Metadaten nicht einfach als Fließtext vorliegen, sondern sowohl die Art eines Metadatums, als auch der Wert festgelegt sind. Die strukturierte Aufnahme der Objekte unterscheidet ein Repositorium maßgeblich von einer einfachen Literaturliste auf der Homepage einer Wissenschaftlerin.

Unter sicherer Speicherung ist in diesem Kontext zu verstehen, dass Metadaten und digitale Objekte vor Speicherverlusten und Manipulation geschützt werden sollen. Je nach Ausprägung des Repositoriums kann das mit unterschiedlichem Aufwand und unterschiedlich weitgehenden Zielen erfolgen, die verlässliche Speicherung ist aber ein wie auch immer umgesetztes Ziel eines jeden Repositoriums.[15] Es gibt keine Repositorien, die lediglich zur temporären Ablage oder als Arbeitsumgebung dienen. Entsprechend wird in der Praxis zum Beispiel zwischen Forschungsdatenrepositorien und virtuellen Forschungsumgebungen differenziert, wobei letztere „potenziell den gesamten Forschungsprozess – von der Erhebung, der Diskussion und weiteren Bearbeitung der Daten bis zur Publikation der Ergebnisse“[16] unterstützen. Die Fokussierung auf digitale Objekte ist es, was ein Repositorium zum Beispiel von einer Knowledge Base abgrenzt, bei der die Information im Vordergrund steht und nicht deren Form.

Unabhängig von der Art der Objekte, die ein Repositorium speichert, seinen Regeln zur Aufnahme neuer Objekte oder für den Zugriff auf gespeicherte Objekte kann daher der Begriff *Repositorium* wie folgt definiert werden: *Ein Repositorium ist ein System zur sicheren Speicherung digitaler Objekte und der die Objekte beschreibenden strukturierten Metadaten.*

Aschenbrenner und Neuroth weisen auf den unterschiedlichen Hintergrund und die voneinander unabhängige Entwicklung verschiedener Repositorien hin. Die sich daraus sowie aus Fokus und Zielgruppe ergebenden unterschiedlichen Funktionen spiegeln sich nach Aschenbrenner und Neuroth auch in der Bezeichnung von

[15] Zum Beispiel stellen einige Repositorien den Erhalt des Bitstroms der einzelnen Objekte sicher, während andere die von einem Objekt repräsentierte Information erhalten wollen. Voraussetzungen, Konzepte und Techniken zum Erhalt digitaler Objekte und/oder der Informationen, die sie repräsentieren, sind unter dem Begriff der Langzeitarchivierung Thema aktueller Forschung.

[16] Definition Virtuelle Forschungsumgebung der Arbeitsgruppe Virtuelle Forschungsumgebungen in der Schwerpunktinitiative Digitale Information der Allianz der deutschen Wissenschaftsorganisationen: http://www.allianzinitiative.de/de/handlungsfelder/virtuelle-forschungsumgebung.html, abgerufen am 06.06.2014.

Repositorien wider: „Repositorien-Systeme decken je nach Fokus und Zielgruppe unterschiedliche Funktionen ab, die sich oft auch in spezifischen Bezeichnungen spiegeln (z.B. ‚institutional repositories' für Publikationsserver, ‚trusted repositories' für Langzeitarchivierungsumgebungen, oder ‚open access repositories' für frei zugängliche Daten)"[17]. Entsprechend können – von der hier vorgeschlagenen allgemeinen Definition eines Repositoriums ausgehend – spezielle Repositorienformen definiert werden. Im Umfeld von wissenschaftlichen Bibliotheken beschreiben die Begriffe „Forschungsdatenrepositorium" oder „Open-Access-Repositorium" Repositorien, die spezialisiert sind auf die Aufnahme von Forschungsdaten beziehungsweise von Objekten, zu denen freier Zugang im Sinne der Open-Access-Bewegung gewährt wird.

Die meisten Repositorien bieten über die obige Definition hinausgehende Funktionen an. Die reine Speicherung zum Selbstzweck, ohne jegliche Möglichkeiten zum Export und somit zur Nachnutzung der Objekte, ist in der Regel nicht das Ziel eines Repositoriums. Der Zugriff auf gespeicherte Objekte ist in der obigen Definition bewusst außen vor gelassen worden, da es durchaus auch Repositorien gibt, die einen Zugriff auf die gespeicherten Objekte nicht zulassen. Hintergrund kann zum Beispiel sein, digitale Objekte der Langzeitarchivierung zuführen zu wollen, auch wenn sie aus rechtlichen Gründen mehrjährigen Sperrfristen unterliegen. Gerade für den wissenschaftlichen Diskurs ist der Zugriff auf die gespeicherten Objekte und das zuverlässige Referenzieren digitaler Objekte von besonderer Bedeutung. Viele Repositorien bieten entsprechende Funktionen, zum Beispiel das Auffinden digitaler Objekte unter Nutzung eines Persistent Identifiers, den Export von Zitationsangaben in BibTeX oder RIS oder auch RSS-Feeds. Eine Liste mit einigen Kernfunktionalitäten von Repositorien findet sich in Aschenbrenner und Neuroth 2011. Das DINI-Zertifikat für Dokumenten- und Publikationsservices (DINI-Zertifikat) hat unter anderem „die Definition von Mindestanforderungen an Dokumenten- und Publikationsservices sowie deren detaillierte Beschreibung" und „das Aufzeigen aktueller zukünftiger Entwicklungstendenzen bei der Gestaltung von Services und beim Austausch von Informationen"[18] zum Ziel. Dementsprechend beschreibt es eine Reihe von Funktionen, die von Repositorien im Umfeld von Dokumenten- und Publikationsservices üblich sind.

Die Anzahl verschiedener Spezialrepositorien ist letztlich unermesslich. Eine vollständige Auflistung aller existierender speziellen Repositorien und ihrer Funktionen ist daher weder möglich noch sinnvoll. Abschließend sollte nur noch darauf hingewiesen werden, dass Repositorien, die die Weitergabe der gespeicherten Objekte und ihrer Metadaten nicht vorsehen, eigentlich als Spezialfall angesehen werden

[17] Aschenbrenner und Neuroth 2011, S. 102f.
[18] DINI 2011, S. 10.

können. Der Großteil der Systeme, die heutzutage als Repositorium bezeichnet werden, sieht eine Weitergabe der Objekte und ihrer Metadaten prinzipiell vor, auch wenn es zum Teil Zugriffsbeschränkungen auf alle oder einzelne Objekte gibt.

Diese Arbeit setzt sich mit dem Zugang zu und dem Austausch von Metadaten und digitalen Objekten mit und zwischen Repositorien auseinander. Daher grenzen wir hier unsere allgemeine Definition von Repositorien ein, indem wir zusätzlich die Weitergabe der gespeicherten Informationen fordern. Für den weiteren Verlauf dieser Arbeit definieren wir ein Repositorium wie folgt: *Ein Repositorium ist ein System zur sicheren Speicherung und Weitergabe digitaler Objekte und der die Objekte beschreibenden strukturierten Metadaten.*

2.2 Austausch digitaler Objekte zwischen Repositorien

2.2.1 Benutzeroberfläche und Schnittstellen

Als eines der ersten Repositorien kann arXiv[19] angesehen werden, dass heutzutage als bekanntester Open-Access-Dokumentenserver gilt. Paul Ginsparg beschreibt in seinem Artikel *First steps towards electronic research communication* aus dem Jahr 1994, wie es zur Entwicklung von arXiv[20] kam. Laut Ginsparg gab es in seinem Fachgebiet der theoretischen Hochenergiephysik mindestens seit Mitte der 1970er Jahre eine ausgeprägte „preprint culture“, bei der neue Ideen und Ergebnisse durch den Versand von ausgedruckten Preprints verbreitet wurden. Fortschritte in Bezug auf die Technik (Entstehen von TEX, Verfügbarkeit bezahlbarer Workstations und das exponentielle Wachstum der Vernetzung dieser Computer) „rendered the development of e-print archives ‚an accident waiting to happen‘.“[21] Die Einrichtung eines „e-print archives“ wurde laut Ginsparg motiviert, durch die Chance den wissenschaftlichen Austausch durch Nutzung der technischen Fortschritte zu vereinfachen. Als Ergebnis wurde dieser Austausch günstiger und schneller. Man kann somit festhalten, dass bei arXiv der Austausch von wissenschaftlichen Publikationen im Vordergrund stand und damit insbesondere der Zugriff auf beziehungsweise der Export der digitalen Objekte.

[19] http://arxiv.org, abgerufen am 06.06.2014.

[20] Der Artikel beschreibt die Entstehung von hep-th@xxx.lanl.gov, das 1999 in arXiv umbenannt wurde und unter http://arxiv.org gefunden werden kann.

[21] Ginsparg 1994, S. 391.

Das von Ginsparg beschriebene Repositorium nutzte vor allem E-Mails zur Aufnahme, aber auch zum Export der gespeicherten Objekte. Die dominante Rolle, die Weboberflächen in Bezug auf den Zugriff auf Repositorien bekommen sollten, sah Ginsparg bereits voraus: „Although the WorldWideWeb still represents only a small fraction of the overall usage, this access mode is expected to become dominant in the near future.“[22] Bis heute erfolgt die Nutzung eines Repositoriums in der Regel über eine Weboberfläche. Für die direkte Nutzung hat sich dieser Weg durchgesetzt. Soll die Nutzung der Daten in einem Repositorium jedoch indirekt erfolgen, das heißt unter Verwendung anderer Services oder zwischengeschalteter Software, so ist eine Weboberfläche nicht von großem Nutzen. Die zwischengeschalteten Services oder Software müssen die Daten in einer maschineninterpretierbaren Art aus dem Repositorium erhalten. So ist zum Beispiel die Formatierung eines Titels einer Publikation als Überschrift nicht hilfreich, vielmehr wird die Angabe gebraucht, dass es sich um den Titel und nicht zum Beispiel um eine Zusammenfassung handelt. Diese Art Daten im Internet über Standardprotokolle und -formate bereitzustellen, ist die Grundidee des Semantic Web. Wahrscheinlich auf Grund der zeitlichen Abläufe bei der Entwicklung von Repositorien und der Umsetzung des Semantic Web haben sich im Umfeld von Repositorien jedoch zwei spezielle Schnittstellen für den Austausch von Daten durchgesetzt, die den Ansätzen des Semantic Web nicht folgen. Zum einen kann das Open Archive Initiative – Protocol for Metadata Harvesting[23] (OAI-PMH) als „de-facto-Standard“ im Umfeld von Repositorien angesehen werden.[24] Zum anderen setzte sich in den letzten Jahren immer mehr das Simple Web-service Offering Repository Deposit[25] (SWORD) durch.

Zum Beispiel nutzt die Deutsche Nationalbibliothek OAI-PMH zur automatisierten Ablieferung von Netzpublikationen, wobei die Metadaten eine „Adresse der elektronischen Ressource zur Ablieferung“ und eine „Adresse der elektronischen Ressource“ als obligatorische Felder enthalten müssen.[26] Über die im erstgenannten Feld übermittelte Adresse werden die Objekte anschließend von der Deutschen Nationalbibliothek geladen. Die OAI-PMH-Schnittstelle ist eine grundlegende Voraussetzung zum Beispiel für Dienste wie die Bielefeld Academic Search Engine BASE[27], eine Suchmaschine für akademische Open-Access-Publikationen.

SWORD dient dem Einstellen digitaler Objekte in ein Repositorium. Eine solche Schnittstelle ist Voraussetzung für die Einbindung von Repositorien in externe Programme, zum Beispiel um digitale Objekte direkt aus der Arbeitsumgebung, in

[22] Ginsparg 1994, S. 393.
[23] http://www.openarchives.org/pmh/, abgerufen am 06.06.2014.
[24] Das DINI-Zertifikat bezeichnet OAI-PMH als de-facto-Standard: DINI 2011, S. 47.
[25] http://swordapp.org, abgerufen am 06.06.2014.
[26] DNB 2012, S. 5.
[27] http://www.base-search.net, abgerufen am 06.06.2014.

der sie entstehen, in ein Repositorium einbringen zu können. Es gibt inzwischen allerdings auch Repositoriensoftware, die neben einer SWORD-Schnittstelle selbst auch einen SWORD-Client beinhaltet. So können Objekte von einem Repositorium in ein anderes eingebracht und damit übertragen oder kopiert werden. Analog dazu gibt es Repositoriensoftware, die einen OAI-PMH-Client integriert hat, um Metadaten und digitale Objekte aus anderen Repositorien übernehmen zu können.[28]

Aus einer praktischen Perspektive heraus kann man zwei Unterschiede zwischen OAI-PMH und SWORD benennen: OAI-PMH dient dazu, Daten aus Repositorien zu ziehen, wohingegen SWORD zum Einbringen von Daten in ein Repositorium verwendet wird. Betrachtet man beide Protokolle in Bezug auf die Interaktion zwischen Repositorien – ausgehend von Repositoriensoftware, die nicht nur Schnittstellen, sondern auch Clients beinhaltet, – so unterscheiden sie sich vor allem darin, wer den Austausch initiiert. Bei SWORD wird der SWORD-Client des exportierenden Repositoriums aktiv, während bei OAI-PMH der OAI-PMH-Client des importierenden Repositoriums die Verbindung initiiert und Daten abfragt.

2.2.2 Metadaten außerhalb von Repositorien

OAI-PMH und SWORD sind im Umfeld von Repositorien bekannte Schnittstellen. Doch über den Kontext von Repositorien hinaus sind sie eher unbekannt. Für die Nutzer von Repositorien spielen die beiden Protokolle nur eine indirekte Rolle: Sie sind für die Umsetzung von Software und Diensten Voraussetzung, werden jedoch nie direkt genutzt. Viele Repositorien bieten aber auch direkt den Export beziehungsweise Download von Metadaten an, wozu in der Regel Formate zum Einsatz kommen, die der Literaturverwaltung entstammen: BibTeX und RIS. Während BibTeX und RIS für die Literaturverwaltung und Generierung von Zitationen aus Metadaten hilfreich sind, können sie nicht als allgemeine Lösung zur Nachnutzung der in Repositorien gespeicherten Metadaten angesehen werden. Gerade in Bezug auf BibTeX gibt es immer wieder große Unterschiede, was die genutzten Felder angeht. Darüber hinaus kann man Repositorien nicht mit Dokumentenservern gleichsetzen, oder gar davon ausgehen, dass ausschließlich Literatur in Repositorien gespeichert wird. Verschiedene Repositorien enthalten sehr unterschiedliche Metadaten, die sich zum Teil nicht in BibTeX oder RIS darstellen lassen.

Auch im Austausch von Metadaten über das direkte Umfeld von Repositorien hinaus helfen OAI-PMH und SWORD kaum weiter. SWORD wurde direkt für das Einbringen von Daten in Repositorien entwickelt und ist ausschließlich im

[28] DSpace (http://www.dspace.org, abgerufen am 06.06.2014) ist ein Beispiel für Repositoriensoftware, die sowohl einen OAI-PMH- als auch einen SWORD-Client integriert hat.

Repositorienumfeld verbreitet.[29] Als Google Sitemaps einführte, um die Indexierung von Websites zu verbessern, wurden auch OAI-PMH-Schnittstellen als Sitemaps unterstützt. Im Jahr 2008 stellte Google den Support von OAI-PMH jedoch ein. Als Grund wurde angegeben, dass die erhaltenen Informationen nicht im Verhältnis zu der Anzahl der Quellen stünde, die auf eine Unterstützung von OAI-PMH durch Google angewiesen seien. Von Mai 2008 an setzte Google ganz auf XML-Sitemaps und unterstützte OAI-PMH nicht mehr.[30] Der Nachweis der eigenen Publikationen in Google Scholar ist für viele Wissenschaftler von großer Bedeutung, so dass viele Dokumentenserver eine XML-Sitemap anbieten und in HTML über meta-Tags Metadaten wie Titel, Autor und dergleichen einbinden.

Hinzu kommt, dass OAI-PMH und SWORD spezielle Schnittstellen sind. OAI-PMH dient zwar dem Abruf von Metadaten, eine freie Suche über die Metadaten von Objekten eines Repositoriums ist aber zum Beispiel nicht möglich. Lediglich die Angabe eines Zeitraums ist möglich, so dass nur Metadaten von Objekten übertragen werden, die im angegebenen Zeitraum in das Repositorium eingestellt oder geändert wurden. OAI-PMH bietet den Abruf der Metadaten aller Objekte oder sogenannter Sets an. Ein Set ist eine Menge von digitalen Objekten des Repositoriums, die vom Administrator des Repositoriums nach beliebigen Kriterien zusammengestellt wurde. Es gibt keine verbindlichen Regeln darüber, welche Sets angeboten werden müssen. Auch in Bezug auf die verwendeten Metadatenschemata sind Repositorien mit einer OAI-PMH-Schnittstelle relativ frei: OAI-PMH selbst definiert nicht, welchem Schema die Metadaten folgen müssen, vielmehr unterstützt es beliebige Metadatenschemata, solange sie sich als XML übertragen lassen. Dafür gibt es eine Funktion, mit der alle vom Repositorium unterstützten Metadatenschemata über die OAI-PMH-Schnittstelle abgefragt werden können. Bei einer Anfrage nach Metadaten muss das gewünschte Schema angegeben werden. Theoretisch können verschiedene Repositorien über OAI-PMH-Schnittstellen ihre Daten in komplett unterschiedlichen Metadatenschemata anbieten. Um ein Mindestmaß an Kompatibilität zu erreichen, wurde von OAI festgelegt, dass Repositorien als eines der Metadatenschemata Dublin Core ohne Qualifier anbieten müssen,[31] aber darüber hinaus gibt es keine verbindlichen Regeln.

Diese Mängel werden auch im DINI-Zertifikat benannt: „Das OAI-Protokoll bietet Interoperabilität auf einem hinsichtlich der zu erfüllenden Anforderungen

[29] Vgl. zur Entwicklung von SWORD http://swordapp.org/about/a-brief-history und zur Verbreitung http://swordapp.org/sword-v1/sword-v1-implementations und http://swordapp.org/sword-v2/sword-v2-implementations, alle abgerufen am 06.06.2014.

[30] Die offizielle Ankündigung von Google die Unterstützung von OAI-PMH einzustellen fand als Blogeintrag statt, der bis heute abrufbar ist: http://googlewebmastercentral.blogspot.de/2008/04/retiring-support-for-oai-pmh-in.html, abgerufen am 06.06.2014.

[31] Vgl. http://www.openarchives.org/OAI/openarchivesprotocol.html, abgerufen am 06.06.2014.

sehr niedrigen Niveau. [...] Um qualitativ hochwertige Dienste aufbauen zu können, die auf der Nutzung über das OAI-Protokoll eingesammelter Daten basieren, sind zusätzliche Festlegungen sinnvoll, die die großen Freiräume, die die reine Protokollspezifikation lässt, ausfüllen."[32] Dazu enthält das DINI-Zertifikat Regeln, die zum Beispiel bestimmte Sets für Repositorien vorschreiben, die das DINI-Zertifikat erhalten wollen. Das DINI-Zertifikat adressiert bewusst Anbieter von Dokumenten- und Publikationsservices, so dass die Regeln für die OAI-PMH-Schnittstelle in diesem Kontext gesehen werden müssen und nicht generell für beliebige Repositorien verallgemeinert werden können.

Sollen Daten mittels OAI-PMH aus einem Repositorium exportiert werden, muss also ein OAI-PMH-Client implementiert oder eine entsprechende Library genutzt werden. Des Weiteren muss das Repositorium, das die gewünschten Daten enthält, hinsichtlich der angebotenen Metadatenschemata und gegebenenfalls der angebotenen Sets untersucht werden. Einen generischen Zugang, der, einmal implementiert, für eine große Menge verschiedener Datenquellen genutzt werden kann und unter Umständen auch noch Informationen über die Metadaten an sich liefert, gibt es im Bereich von Repositorien bislang nicht. Zumindest bis sich ein entsprechender Standard gebildet und durchgesetzt hat, bleiben OAI-PMH und SWORD wichtige Übergangstechnologien im Bereich von Repositorien, die überhaupt eine Möglichkeit für den Austausch von Metadaten und digitalen Objekten bieten.

2.3 Öffnen der Datensilos

2.3.1 Probleme des Informationsaustausches

Um die technischen Probleme beim Austauschen von Metadaten und digitalen Objekten zu verdeutlichen, wollen wir als fiktives Beispiel die Erstellung einer Universitätsbibliographie betrachten. Es soll also eine Liste aller Publikationen der an einer Universität in einem bestimmten Jahr beschäftigten Wissenschaftler erstellt werden. Als Datenquellen bieten sich in unserem Beispiel der Bibliothekskatalog der Universitätsbibliothek, der Dokumentenserver der Universität und die öffentlich zugänglichen Internetseiten der Wissenschaftler an. Das Rechenzentrum der Universität betreibt ein Content Management System (CMS) für die Homepages der Wissenschaftler und ein zugehöriges Modul zur Pflege einer Bibliographie, dieses Angebot wird aber nicht von allen Wissenschaftlern genutzt. Von der Universi-

[32] DINI 2011, S. 47f.

tätsbibliothek bekommen wir den Katalog als eine große Datei in MAB2[33]. Der Dokumentenserver integriert eine OAI-PMH-Schnittstelle, die als Format Dublin Core[34] und XMetaDissPlus[35] nutzt. Aus dem Bibliographie-Modul des eingesetzten CMS können die Publikationslisten als BibTeX-Datei heruntergeladen werden, zumindest die der Wissenschaftler, die das Angebot des Rechenzentrums nutzen. Auf die Homepages der Wissenschaftler, die das angebotene CMS nicht nutzen, kann allenfalls mit einem Web-Crawler zugegriffen werden, wobei versucht werden muss, anhand von Stichworten die Seite zu finden, die gegebenenfalls die Bibliographie enthält. Diese Bibliographien können in beliebigen Formaten oder einfach als Text auf einer einfachen HTML-Seiten vorliegen. Jede dieser Datenquellen sieht eine eigene Zugriffsart (Dateizugriff, OAI-PMH-Client, HTTP-Client, Web-Crawler) vor und bietet ihre Daten in einem eigenen Format (MAB2, Dublin Core oder XMetaDissPlus, BibTeX und HTML) an. Entsprechend aufwändig ist die Erstellung der Universitätsbibliographie.

Die Nutzung spezieller APIs und Formate erschwert den Zugang zu den Informationen. Jede spezielle Schnittstelle, die einen eigenen Client erfordert, jedes spezielle Format, das unterstützt werden muss, bringt zusätzlichen Entwicklungsaufwand mit sich. Es gibt viele Gründe für die unbefriedigende Format- und Schnittstellenvielfalt. In der Regel ist sie die Folge unabhängig verlaufender Entwicklungen, also von Entwicklungen die verschiedenen Anforderungen unterliegen und unterschiedliche Ziele haben.

Noch schwieriger als der Umgang mit Daten unterschiedlicher Formate ist der Umgang mit Daten, die keinem festen Format unterliegen, oder zumindest keinem Format, das die Semantik der Daten wiedergibt. Im Beispiel sind dies Bibliographien auf den Homepages von Wissenschaftlern, die in HTML vorliegen, jedoch nicht in einem Format, das den Titel einer Publikation, das Erscheinungsjahr und so weiter als solche Informationen kennzeichnet. Menschen erkennen aus Zusammenhängen Semantik. Sie erwarten Metadaten von Publikationen in einer Bibliographie, sie erkennen einen Autorennamen, der von einem hervorgehobenen Titel gefolgt wird und können unterscheiden, ob eine Jahreszahl in einer Publikationsangabe Teil

[33]MAB steht für Maschinelles Austauschformat für Bibliotheken. MAB ist über lange Zeit vor allem in deutschen Bibliotheken eingesetzt worden und wird nun durch MARC, ein international verbreitetes Katalogisierungsformat, ersetzt. Vgl. http://www.dnb.de/DE/Standardisierung/Formate/MAB/mab_node.html, abgerufen am 06.06.2014.

[34]Dublin Core ist ein weit verbreitetes Metadatenschema und wird inzwischen auch als RDF Vokabluar bereitgestellt, siehe http://www.dublincore.org, abgerufen am 06.06.2014.

[35]XMetaDiss wurde ursprünglich zum Austausch von Metadaten über Hochschulschriften verwendet. XMetaDissPlus ist eine Erweiterung von XMetaDiss, die von der Deutschen Nationalbibliothek als ein Format zur automatisierten Abgabe von Netzpublikationen akzeptiert wird. Vgl. http://www.dnb.de/DE/Standardisierung/Formate/MAB/mab_node.html abgerufen am 06.06.2014 und DNB 2012.

eines Titels oder eine Angabe zum Erscheinungsjahr ist. Computer erkennen solche Zusammenhänge bislang nicht ohne weiteres.

2.3.2 Generischer Zugang zu Informationen

In ihrem viel zitierten Artikel *The Semantic Web* adressieren Tim Berners-Lee und seine Co-Autoren die im Abschnitt 2.3.1 genannten Probleme. Als Ursache benennen sie die Ausrichtung von Informationen auf Menschen als Konsumenten: „Information varies along many axes. One of these is the difference between information produced primarily for human consumption and that produced mainly for machines. [...] To date, the Web has developed most rapidly as a medium of documents for people rather than for data and information that can be processed automatically.“[36] Die Lösung sehen sie in einer Erweiterung des World Wide Web zum *Semantic Web*, in dem Informationen auch maschineninterpretierbar angeboten werden. Darüber hinaus sollen in *Ontologien*[37] Zusammenhänge zwischen verschiedenen Begriffen und Regeln bereitgestellt werden, die automatisierte logische Schlussfolgerungen ermöglichen.

Als Erweiterung des bestehenden World Wide Web nutzt das Semantic Web die gleiche Zugangsart, die auch jetzt schon im Internet verbreitet ist: HTTP(S) in Verbindung mit DNS (zur Auflösung von URIs auf IP-Adressen). Arbeitet man mit verschiedenen Datenquellen, die als Teil des Semantic Web realisiert wurden, so ist die Art des Datenzugriffes und -formats damit einheitlich geklärt: „The use of HTTP as a standardized data access mechanism and RDF as a standardized data model simplifies data access compared to Web APIs, which rely on heterogenous data models and access interfaces.“[38] Berners-Lee, Hendler u. a. zeigen auf, dass Ontologien auch zur Lösung im Umgang mit der Formatvielfalt beitragen können: „A program that wants to compare or combine information across the two databases has to know that these two terms [two different terms for zip code] are being used to mean the same thing. Ideally, the program must have a way to discover such common meanings for whatever databases it encounters. A solution to this problem is provided by the third basic component of the Semantic Web, collections of information called ontologies.“[39] Die Daten im Semantic Web werden in RDF bereitgestellt. Darüber hinaus helfen Ontologien einzelne Ausdrücke in Relation zu setzen und so die Semantik der Ausdrücke in eine Form zu bringen, mit der Computer arbeiten können: „The computer doesn't truly ‚understand' any of this

[36] Berners-Lee, Hendler u. a. 2001, S. 37.
[37] Vgl. Fußnote 9 auf S. 8.
[38] Bizer, Heath u. a. 2009, S. 4.
[39] Berners-Lee, Hendler u. a. 2001, S. 40.

information [classes of objects, relations among them and the semantic of inference rules], but it can now manipulate the terms much more effectively in ways that are useful and meaningful to the human user.“[40]

Repositorien und das Semantic Web können sich gegenseitig bereichern. Repositorien enthalten wichtige Inhalte, die für das Semantic Web von Interesse sind. Damit sich aus der Idee des Semantic Web Dienste entwickeln, die einer breiten Masse von Menschen dienlich sind, braucht es zunächst interessante Daten. Auf der anderen Seite erfolgt die Speicherung von Informationen in Repositorien nicht zum Selbstzweck.[41] Vielmehr bieten die meisten Repositorien gezielt Funktionen zum einfachen Zugriff auf die gespeicherten Inhalte und dienen damit der Informationsdistribution. Das Bereitstellen der gespeicherte Daten im Semantic Web ist daher als eine wichtige Funktion von Repositorien anzusehen. Methoden und Techniken des Semantic Web können dabei helfen, die bisherigen Probleme der Repositorien beim Austausch von Informationen zu lösen. Ontologien können helfen, Metadaten zwischen verschiedenen Metadatenschemata zu konvertieren. Der vereinheitlichte Zugriff über HTTP(S) und RDF kann dafür sorgen, dass Informationen aus Repositorien in andere Systeme eingebunden werden können, ohne dass auf spezielle Schnittstellen zurückgegriffen werden muss.

Durch die wachsende Bedeutung des Semantic Web entstehen immer mehr Programme und Programmbibliotheken, die den Aufwand zur Nutzung und Verarbeitung von Informationen aus dem Semantic Web sinken lassen. Werden bei der Einbindung von Repositorien in das Semantic Web auch Informationen über den Zeitpunkt der Erstellung oder letzten Änderung von Metadaten bereitgestellt, so kann vielleicht eines Tages auf eine OAI-PMH-Schnittstelle und damit auf ihre Programmierung, Pflege und die Programmierung von OAI-PMH-Clients verzichtet werden.

2.3.3 Repositorien ins Semantic Web einbinden

Das Semantic Web soll Daten auch maschineninterpretierbar bereitstellen. Repositorien speichern Metadaten strukturiert ab, das heißt, in Repositorien stehen Metadaten bereits in maschineninterpretierbarer Form zur Verfügung. Zum Beispiel gibt es in einem Dokumentenserver ein oder mehrere Felder, worin die Namen der Autorinnen und Autoren einer Publikation gespeichert sind. Auf der Übersichtsseite einer Publikation wird der Inhalt dieser Felder vielleicht dem Titel der Publikation voran und kursiv gesetzt. Die Information darüber, dass es sich dabei um die

[40] Berners-Lee, Hendler u. a. 2001, S. 40.

[41] Vgl. S. 15. Alle Verweise auf Seitenzahlen ohne weitere Literaturangabe beziehen sich hier und im Folgenden auf die vorliegende Arbeit.

Autorinnen und Autoren der Publikation handelt, ist jedoch vorhanden, sie müsste nur mit ausgegeben werden.

Das Vorhandensein solcher Informationen, die strukturierte Aufnahme von Metadaten, ist ein wichtiges Argument, warum gerade Repositorien ins Semantic Web eingebunden werden sollen: Die Daten sind bereits vorhanden, es geht darum sie (besser) nutzbar zu machen. Repositorien enthalten Metadaten, die in der Regel manuell erstellt und oft von Bibliothekaren korrigiert und ergänzt wurden. Zum Teil werden auch Normdaten genutzt, die meisten Repositorien orientieren sich an bestimmten Metadatenschemata. Wir können also von qualitativ hochwertigen Metadaten ausgehen.[42] Die Herausforderung liegt in der Frage, wie die in Repositorien vorhandenen Daten bereitgestellt und wie sie mit bestehenden Ressourcen des Semantic Web verknüpft werden können. Hierin unterscheiden sich Repositorien maßgeblich von anderen Bereichen, in denen zunächst Daten und Beschreibungen generiert beziehungsweise manuell erstellt werden müssen.

Daten aus Repositorien unter Nutzung von HTTP, RDF und SPARQL bereitzustellen, würde Zugriffsbarrieren einreißen, die der exklusive Zugang über Webschnittstellen für Benutzer und OAI-PMH-Schnittstellen darstellt. Für Repositorien, die die gesammelten Daten zugänglich machen wollen und eine Nachnutzung der gesammelten Daten anstreben, bietet die Bereitstellung ihrer Daten als Linked Data, die Anbindung ihrer Bestände an das Semantic Web eine sehr vielversprechende Chance.

[42] Vgl. Kapitel 1.

Kapitel 3

Repositoriensoftware und Linked Data

Die Idee des Semantic Web ist es, Informationen auch maschineninterpretierbar im Internet bereitzustellen. Hitzler u. a. nennen als Grundvoraussetzung dafür die Notwendigkeit, „einheitliche, offene Standards für die Beschreibung von Informationen zu vereinbaren, die es letztlich ermöglichen sollen, Informationen zwischen verschiedenen Anwendungen und Plattformen auszutauschen und zueinander in Beziehung zu setzen“.[43] Das World Wide Web Consortium (W3C) hat solche Standards mit XML, RDF, RDFS[44], OWL[45] und der Abfragesprache SPARQL geschaffen. In Hitzler u. a. 2008 werden diese Standards umfassend dargestellt, so dass hier nicht im Detail darauf eingegangen werden soll. Der Begriff Linked Data beinhaltet jedoch mehr, als Daten unter Nutzung dieser Standards bereitzustellen: „The term [Linked Data] refers to a style of publishing and interlinking structured data on the Web.“[46] Neben Standards gibt es eine Reihe von Konventionen, die eingehalten werden sollten, wenn Daten als Linked Data bereitgestellt werden.

Repositorien in Richtung des Semantic Web zu öffnen, erfordert die Bereitschaft und Mitarbeit von Repositorienbetreibern. Je einfacher es für sie ist, den Inhalt des von ihnen betriebenen Repositoriums als Linked Data zu veröffentlichen, desto größer wird die Bereitschaft dazu sein. Je mehr Änderungen eine solche Öffnung mit sich bringt, um so aufwändiger dürfte sie für Repositorienbetreiber sein. Neue Arbeitsabläufe, der Verzicht auf im Repositorienumfeld etablierte Schnittstellen und große Softwareumstellungen sollten vermieden werden, wenn es lediglich darum

[43] Hitzler u. a. 2008, S. 11.
[44] Resource Description Framework Schema.
[45] Web Ontology Language.
[46] Bizer, Cyganiak u. a. 2007.

geht, vorhandene Daten zusätzlich auf neue Art und Weise bereitzustellen: Durch Nutzung von für das Semantic Web entwickelten Standards und unter Einhaltung von Konventionen aus dem Linked-Data-Umfeld.

Um Datenmigrationen beim Öffnen bestehender Repositorien in Richtung des Semantic Web zu vermeiden, sollte bestehende Repositoriensoftware erweitert werden, anstatt eine neue Repositoriensoftware zu entwickeln, die auf Triple Stores oder anderer für das Semantic Web entwickelter Software basiert. Neben Konventionen für Linked Data müssen daher auch Architektur und Funktionen bestehender Repositoriensoftware sowie Konventionen aus dem Umfeld von Repositorien berücksichtigt werden.

In diesem Kapitel werden Konventionen des Semantic Web aufgezeigt und in Einklang mit Gegebenheiten aus dem Umfeld von Repositorien gebracht. Es wird ein Überblick über Softwarelösungen für Repositorien gegeben und darüber, wie diese Metadaten und digitale Objekte bereitstellen. Es wird untersucht, ob und gegebenenfalls wie digitale Objekte und Metadaten bereits als Linked Data ins Semantic Web integriert werden. Es geht um die Beantwortung folgender zentraler Fragen, bevor ein Konzept zur Integration von Repositorien in das Semantic Web erstellt werden kann: Wie werden Daten im Semantic Web publiziert? Wie werden Metadaten und digitale Objekte bislang in Repositorien bereitgestellt? Welche Charakteristika von Repositorien sind in einem Konzept zur Einbindung von Repositorien ins Semantic Web zu berücksichtigen?

3.1 Linked Data principles

Tim Berners-Lee veröffentlichte Hintergründe zu Standards und persönliche Ansichten zum Internet in Form von „personal notes“.[47] In einer solchen Mitteilung setzt er sich mit dem Begriff Linked Data auseinander und stellt klar, dass der Begriff Linked Data über die reine Verwendung der genannten Standards hinaus geht: „The Semantic Web isn't just about putting data on the web. It is about making links [...] With linked data, when you have some of it, you can find other, related, data.“[48] Er führt vier Regeln ein, wobei er betont, dass es sich eher um Konventionen handelt. Gegen sie zu verstoßen würde weniger ernste Konsequenzen haben, als viel mehr eine verpasste Gelegenheit darstellen, Daten miteinander zu verknüpfen. Bizer, Heath und Berners-Lee weisen in *Linked Data – The Story So*

[47] http://www.w3.org/DesignIssues/, abgerufen am 06.06.2014.
[48] Berners-Lee 2006.

Far darauf hin, dass diese Regeln als „Linked Data principles“ bekannt geworden sind.[49] Sie lauten wie folgt:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names
3. When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)
4. Include links to other URIs. [sic!] so they can discover more things.[50]

Die folgenden vier Abschnitte befassen sich jeweils mit einem dieser Grundsätze. Dabei wird auf Besonderheiten eingegangen, die sich in Bezug auf den jeweiligen Grundsatz im Zusammenhang mit Repositorien ergeben.

3.1.1 Nutzung von URIs

Die Nutzung von URIs als Name für Objekte ist tief in RDF verankert. Sie ist eine der Grundlagen des Semantic Web. In Bizer, Cyganiak u. a. 2007 wird näher auf den Unterschied zwischen URIs als Name von Objekten und URIs als Adresse für Informationen im Netz eingegangen. Mit Verweis auf die W3C Technical Architecture Group (TAG)[51] wird auf den Unterschied zwischen *Information Resources* und *Non-Information Resources*[52] hingewiesen: „All the resources we find on the traditional document Web, such as documents, images and other media files, are information resources. But many of the things we want to share data about are not: People, physical products, places, proteins, scientific concepts and so on. As a rule of thumb, all ‚real-world objects‘ that exist outside of the Web are non-information resources.“[53] Im Zusammenhang mit *Content Negotiation*[54] weisen Bizer, Cyganiak und Heath darauf hin, dass für eine Non-Information Resource oft drei URIs genutzt werden:

- ein URI zur Identifizierung der Non-Information Resource,

[49] Bizer, Heath u. a. 2009, S. 2.
[50] Berners-Lee 2006.
[51] http://www.w3.org/2001/tag/, abgerufen am 06.06.2014.
[52] *Information Resource* und *Non-Information Resource* werden hier als feststehende Begriffe verwendet und daher nicht übersetzt.
[53] Bizer, Cyganiak u. a. 2007.
[54] Content Negotiation wird im Folgenden nicht übersetzt, da es sich dabei um eine Technik handelt, die auch in der deutschsprachigen Literatur als Content Negotiation bezeichnet wird.

- ein URI einer Information Resource mit einer Repräsentation in RDF/XML und
- ein URI einer Information Resource mit einer Repräsentation in HTML.

Wird der URI zur Identifizierung des Objekts dereferenziert, so wird mittels Content Negotiation entschieden, auf welchen der beiden anderen URIs verwiesen wird.

In Repositorien werden digitale Objekte und Metadaten gespeichert. Die gespeicherten Objekte selbst sind gemäß dem obigen Zitat Information Resources. Die von Bizer, Cyganiak und Heath beschriebene Nutzung dreier URIs für eine Non-Information Resource lässt sich jedoch auch für eine Information Resources anwenden.

Man mag sich die Frage stellen, ob es sinnvoll ist, eine Information Resource wie eine Non-Information Resource zu behandeln. Betrachten wir das an einem Beispiel: In einem Repositorium, in unserem Beispiel ein Dokumentenserver für wissenschaftliche Publikationen, gibt es mehrere verschiedene URIs, die in Bezug zu einem im Repositorium gespeicherten wissenschaftlichen Artikel gebraucht werden:

- Ein URI kann im wissenschaftlichen Diskurs verwendet werden, zum Beispiel um in einem anderen Artikel auf den im Repositorium gespeicherten Artikel zu verweisen. Dieser URI dient demnach als Identifikator des im Repositorium gespeicherten Artikels.
- Es werden URIs für Seiten benötigt, die Informationen wie den Autor, den Titel und eine Zusammenfassung enthalten sowie Links auf alle vorliegenden Repräsentationen und weitere zugehörige Materialien. Dazu braucht es je einen URI für die Repräsentation dieser Informationen in HTML und in RDF/XML.
- Des Weiteren wird je ein URI für jede Repräsentation des Artikels gebraucht, zum Beispiel für den Artikel als PDF, als PostScript und die zugehörigen Latex-Quellen.

Der URI, der als Identifikator des Artikels dient, verweist unter Nutzung von Content Negotiation entweder auf den URI mit Informationen in HTML oder auf den URI mit Informationen in RDF/XML, die Links auf alle Repräsentationen des Artikels enthalten. Dementsprechend ist es sinnvoll ein in einem Repositorium gespeichertes digitales Objekt so zu behandeln, wie es Bizer, Cyganiak und Heath für eine Non-Information Resource vorschlagen, da ein solches Objekt in unterschiedlichen Kontexten referenziert wird. Hinzu kommt, dass Repositorien darauf

ausgelegt sind, verschiedene Repräsentationen eines solchen Objekts bereitzuhalten, und dass je Repräsentation ein URI gebraucht wird.

Die TAG weist in ihrem Entwurf „Dereferencing HTTP URIs“[55] darauf hin, dass HTTP-Statuscodes keine Rückschlüsse auf die Art der Ressource zulassen. Lediglich ein HTTP-Statuscode 200 ist ein eindeutiger Hinweis, dass es sich bei der Ressource, deren URI dereferenziert wurde, um eine Information Resource handelt. Ein HTTP-Statuscode 303, wie er bei der Content Negotiation verwendet wird, weist auf eine beliebige Ressource hin. HTTP-Statuscodes aus dem Bereich 4XX und 5XX lassen keine Rückschlüsse auf die Ressource des dereferenzierten URIs zu.[56] Daher spricht nichts dagegen die in Repositorien gespeicherten digitalen Objekte als Non-Information Resource zu betrachten, wenn es um die Vergabe von URIs geht.

Obwohl Repositorien ihre Daten noch nicht für das Semantic Web bereitstellen, ordnen sie in der Regel jedem gespeicherten digitalen Objekt einen URI zu seiner Identifikation, URIs für die einzelnen Dateien und einen URI für Informationen zum Objekt in HTML zu. Im Vergleich zur oben beispielhaft skizzierten Verwendung von URIs fehlt also lediglich ein URI für die Informationen zum Objekt in RDF/XML.

Der vom Repositorium vergebene URI zur Identifikation eines Objekts, wird zur Nutzung für Bookmarks oder Verweise in wissenschaftlichem Kontext empfohlen, meist handelt es sich dabei um einen sogenannten Persistent Identifier in Form eines Digital Object Identifiers (DOI), eines Handles, eines Uniform Resource Names (URN) oder dergleichen. Es bietet sich an, diesen URI auch im Rahmen von Linked Data zur Identifikation des im Repositorium gespeicherten Objekts zu nutzen. Sowohl Berners-Lee, als auch Bizer, Cyganiak und Heath empfehlen keine DOIs, URNs oder der gleichen an dieser Stelle zu nutzen, sondern HTTP-URIs.[57] Hierauf wird im Abschnitt 3.1.2 näher eingegangen. Auch die anderen vom Repositorium vergebenen URIs, sowohl für die Informationen in HTML, als auch für die einzelnen Dateien, sollten im Rahmen von Linked Data entsprechend verwendet werden.

[55] http://www.w3.org/2001/tag/doc/httpRange-14/2007-05-31/HttpRange-14, abgerufen am 06.06.2014.

[56] Der Entwurf der TAG wurde später zugunsten eines anderen Vorschlags zurückgestellt. In Bezug auf Rückschlüsse von HTTP-Statuscodes auf die durch den URI identifizierte Ressource wurde jedoch ein Kompromiss gefunden, der hier entsprechend dargestellt ist. Vgl. http://lists.w3.org/Archives/Public/www-tag/2005Jun/0039.html, abgerufen am 06.06.2014.

[57] Vgl. Berners-Lee 2006 und Bizer, Cyganiak u. a. 2007.

3.1.2 HTTP URIs

In Bezug auf die zweite Regel der Linked Data Principles schreibt Berners-Lee: „The second rule, to use HTTP URIs, is also widely understood. The only deviation has been, since the web started, a constant tendency for people to invent new URI schemes (and sub-schemes within the urn: scheme) such as LSIDs and handles and XRIs and DOIs and so on, for various reasons.“[58] Bei Handles, DOIs und der Gleichen handelt es sich um sogenannte Persistent Identifiers, die vor allem im Repositorienumfeld weit verbreitet sind. Neben Handle und DOI ist in Deutschland auch noch urn:nbn:de: zu erwähnen, das von der Deutschen Nationalbibliothek betrieben und genutzt wird.

Persistent Identifiers kamen auf, als der wissenschaftliche Diskurs zunehmend ins Internet verlagert wurde. Im Umfeld elektronischer Publikationen wurde darüber diskutiert, wie wissenschaftliche Publikationen, die nicht auch gedruckt erscheinen, zitiert werden können. Bis heute werden im Internet veröffentlichte Inhalte als flüchtig angesehen, weil zu wenig Anstrengungen unternommen werden, um genutzte URIs stabil zu halten.[59] Nach wie vor wird oft zu wenig dafür getan, dass dauerhaft die selben Inhalte beim Dereferenzieren eines URIs in einem Browser angezeigt werden. Das sorgt vor allem beim elektronischen wissenschaftlichen Publizieren für Probleme, da Zitate und Referenzen unter Umständen nicht mehr nachverfolgt werden können oder gar andere Inhalte präsentiert werden als zu dem Zeitpunkt, zu dem eine Referenz erstellt wurde.

Persistent Identifiers sollen den Problemen, die durch das Ändern von URIs entstehen, vorbeugen. Dazu wird nicht auf einen URI referenziert, sondern auf einen Persistent Identifier. Der Persistent Identifier kann über einen Resolver-Dienst in den URI umgewandelt werden, unter dem der adressierte Inhalt aktuell zu finden ist. Das DINI-Zertifikat zum Beispiel erklärt die Nutzung von Persistent Identifiern zu einer Mindestanforderung an Dokumenten- und Publikationsservices[60] und begründet dies damit, dass so „der Zugriff auf die Dokumente auch bei Veränderungen an dem verwendeten Softwaresystem oder einer zugrunde liegenden Systematik, sichergestellt [wird]. PIs [Persistent Identifiers] eignen sich insbesondere für das Zitieren elektronischer Publikationen, da sie im Gegensatz zu URLs dauerhaft angelegt sind.“[61] Die TAG hält den Einsatz von Persistent Identifiers für unnötig. Persistenz von Bezeichnern sei in Bezug auf HTTP-URIs keine Frage der Technik, sondern eine Managementfrage.[62]

[58] Berners-Lee 2006.
[59] Vgl. Berners-Lee 1998.
[60] DINI 2011, S. 25.
[61] DINI 2011, S. 41.
[62] W3C TAG 2006.

Tim Berners-Lee nennt zwei Gründe, deretwegen versucht würde, neue URI-Schemata oder URN-Unternamensräume zu etablieren: „Typically, these involve not wanting to commit to the established Domain Name System (DNS) for delegation of authority but to construct something under separate control. Sometimes it has to do with not understanding that HTTP URIs are names (not addresses) and that HTTP name lookup is a complex, powerful and evolving set of standards.“[63] Tatsächlich werden URIs als Bezeichner und URLs als Adressen oft verwechselt. Doch genau wie bei URIs ist die Frage der Persistenz von URLs eine Frage des Management und nicht der Technik: Es gibt keinerlei technische Gründe dafür, dass sich ein HTTP-URL ändert, wenn Inhalte auf einem anderem Server hinterlegt werden oder es eine Softwareumstellung gibt. DNS abstrahiert Domainnamen hinreichend von Servern und HTTP bietet mit Statuscodes aus dem Bereich 3XX Möglichkeiten, URLs weiterzuleiten und so auch alte URLs verfügbar zu halten, selbst wenn Inhalten neue URLs zugewiesen wurden.

Bereits 1998 hat Tim Berners-Lee darauf hingewiesen, dass URIs, wenn sie gut gewählt wurden, unabhängig von der eingesetzten Software sind. Im Zusammenhang mit der Nutzung von Themen in einem URI weist er auf Nachteile hin, wenn URIs später anders verwendet werden sollen und sich Themen oder Schlagworte für Themen im Laufe der Zeit ändern.[64] Die TAG weist noch auf ein maßgebliches praktisches Problem bei der Einführung neuer URI-Schemata hin: Neue Schemata, die nicht auf HTTP und DNS aufsetzen, brauchen Clients, die die vom neuen Schema verwendeten Protokolle beherrschen. „http: provides substantial benefits in term of installed software base, user comprehension, scalability and, if required, security, at very low cost. Anyone developing an alternative approach [...] should consider carefully whether that approach is either isomorphic to http:, or makes covert appeal to http: for its implementation. In either case, this strongly suggests [...] that http: itself would be viable, and therefore a preferred, way forward.“[65]

Ein wichtiges Argument für den Einsatz von Persistent Identifiers ist es, durch sie die Intention auszudrücken, Persistenz zu bieten. Hintergrund ist die Zitierbarkeit digitaler Objekte, die nur gewährleistet ist, wenn Identifiers und adressierte Inhalte lange verfügbar und ohne inhaltliche Änderungen bereitgehalten werden. Die TAG verweist darauf, dass dies eine Frage von Namenskonventionen sei, die auch mit http: erreichbar wäre.[66] Einerseits ist das richtig, andererseits haben sich DOIs in den letzten Jahren in etlichen wissenschaftlichen Disziplinen durch-

[63] Berners-Lee 2006.
[64] Berners-Lee 1998.
[65] W3C TAG 2006.
[66] W3C TAG 2006.

gesetzt.[67] Dabei ist es inzwischen aber unerheblich, ob DOIs mit eigenem URI-Schema doi:<DOI> oder als HTTP-URI http://dx.doi.org/<DOI> angegeben werden. Beide Formen sind für die Angabe von DOIs weithin verbreitet und bieten entsprechenden Wiedererkennungswert.

Letztlich gibt es noch zwei Argumente gegen die Nutzung von Persistent Identifiers, selbst wenn sie in Form von HTTP-URIs verwendet werden. Bizer, Cyganiak und Heath schreiben: „Define URIs in an HTTP namespace under your control, where you actually can make them dereferenceable.“[68] In Bezug auf DOIs kann man darauf verweisen, dass der Vergabe von DOIs ein Vertrag mit einem DOI Registrar vorausgeht, der mit einem Vertrag mit einem Domain Registrar vergleichbar ist. Bizer, Cyganiak und Heath schreiben des Weiteren: „HTTP URIs [...] provide a simple way to create globally unique names without centralized management“[69]. Je mehr Anbieter von Linked Data ihre URIs unterhalb von http://dx.doi.org angeben, desto mehr Daten wären bei einem Ausfall von http://dx.doi.org betroffen. Es gibt inzwischen jedoch auch vielfältige technische Möglichkeiten, den Betrieb häufig genutzter HTTP-URIs abzusichern (man denke nur an Adressen wie http://www.google.com oder http://en.wikipedia.org). Andererseits räumt selbst die TAG ein, dass es zum Beispiel bei vernachlässigter Pflege („lack of maintenance“) zum Verlust einer Domain kommen kann.[70] DOIs sind in wissenschaftlichen Publikationen weit verbreitet. Es gibt daher ein großes und breit gestreutes Interesse am Erhalt von dx.doi.org, das über den Erhalt einer Domain durch eine einzelne Organisation, Firma oder Person weit hinausgeht. Letztlich geht es hier auch nicht um die generelle Nutzung von DOIs für Linked Data, sondern um eine spezielle Lösung für die Einbindung von Repositorien ins Semantic Web.

DOIs als HTTP-URI in der Form http://dx.doi.org/<DOI> als Identifier für Linked Data zu nutzen, wird daher sowohl den Hinweisen des W3Cs gerecht als auch den Forderungen aus dem Umfeld von Repositorien und der wissenschaftlichen Praxis in Bezug auf das Referenzieren digitaler Objekte. Eine DOI in der Form http://dx.doi.org/<DOI> verweist auf einen zentralen DOI-Resolver-Dienst über den jede registrierte DOI unter Nutzung von HTTP und DNS dereferenziert werden kann. Seit April 2011 unterstützt dieser zentrale Resolver auch Content Negotiation, eine Entwicklung, die in Hinblick auf die Nutzung von DOIs für Linked Data erfolgte.[71]

[67] Etliche namhafte Verlage und Zeitschriften akzeptieren und verwenden DOIs als Referenzen und vergeben auch selbst DOIs für ihre Publikationen.

[68] Bizer, Cyganiak u. a. 2007.

[69] Bizer, Cyganiak u. a. 2007.

[70] W3C TAG 2006.

[71] http://www.doi.org/news/DOINewsApr11.html, abgerufen am 06.06.2014.

3.1.3 Dereferenzierung von URIs

Bisher wird der generische Zugang zu Daten oft durch spezifische APIs und/oder Formate verbaut. Das Besondere an Linked Data ist der Versuch, den Austausch von und den Zugriff auf Daten zu vereinheitlichen. Hitzler u. a. bringen es auf den Punkt, wenn sie schreiben, eine Grundvoraussetzung zur Realisierung des Semantic Web sei es „einheitliche offene Standards für die Beschreibung von Informationen zu vereinbaren, die es letztlich ermöglichen sollen, Informationen zwischen verschiedenen Anwendungen und Plattformen auszutauschen und zueinander in Beziehung zu setzen, eine Fähigkeit, die *Interoperabilität* genannt wird.“[72] Mit DNS und HTTP wird für Linked Data auf weit verbreitete, etablierte Standards für die Übertragung von Daten gesetzt. Für die inhaltliche Beschreibung wurde mit RDF ein Datenmodell entwickelt, auf das RDFS und OWL aufsetzen, um Begriffe und Relationen einer Domäne zu definieren und in Form von Ontologien[73] beziehungsweise Vokabularen zu publizieren.

RDF gibt eine Struktur aus Tripeln vor, die dem Muster Subjekt, Prädikat, Objekt folgt, wobei Subjekt und Prädikat immer URIs sind und das Objekt ein URI oder ein Literal sein kann. Ein Prädikat gibt die Relation an, in der Subjekt und Objekt zueinander stehen. URIs werden im Semantic Web zur Identifikation von konkreten Dingen („Non-Information Resources“), Beschreibungen („Information Resources“), Entitätstypen und Relationen verwendet.

Die dritte Regel der Linked Data Principles [74] zielt darauf ab, dass alle diese URIs dereferenzierbar sein sollen und mit Hilfe der dafür geschaffenen Standards entsprechende Informationen bereitgestellt werden. Bei der Dereferenzierung von URIs von Entitätstypen oder Relationen sind die entsprechenden Ontologien in RDFS oder OWL zurück zu gegeben, URIs von konkreten Dingen sollen über einen HTTP-Statuscode 303 auf ihre Beschreibungen verweisen. Bei der Dereferenzierung von URLs zur Beschreibung konkreter Dinge werden die entsprechenden Informationen in RDF erwartet.

Mit der dieser Regel der geht es Berners-Lee jedoch nicht allein darum, dass alle diese URIs dereferenzierbar sind und die genannten Standards verwendet werden. Anhand von Beispielen weist er darauf hin, dass es Projekte gab, in deren Rahmen Linked Data erstellt wurde, die Daten nach Abschluss des Projekts jedoch in einem ZIP-Archiv im Internet angeboten wurden, anstatt direkt in den entsprechenden Standards über HTTP zur Verfügung gestellt zu werden. Des Weiteren nimmt er Bezug auf große Datenmengen, für die oft SPARQL-Endpoints eingerichtet würden,

[72] Hitzler u. a. 2008, S. 11.
[73] Vgl. Fußnote 9 auf S. 8.
[74] Vgl. S. 27.

wobei auch dann die Daten immer auch direkt als RDF verfügbar gemacht werden sollten.[75] Dadurch, dass Informationen direkt als RDF zur Verfügung gestellt werden und über HTTP abrufbar sind, wird die von Hitzler u. a. benannte Interoperabilität erreicht.

3.1.4 Link Linked Data

Bizer, Cyganiak und Heath legen besonderen Wert auf die Bedeutung von Links. Daten zu verlinken bezeichnen sie als einen der beiden Grundsätze von Linked Data, neben der Verwendung von RDF. Weiter schreiben sie: „The basic assumption behind Linked Data is that the value and usefulness of data increases the more it is interlinked with other data. In summary, Linked Data is simply about using the Web to create typed links between data from different sources.“[76] RDF-Statements, die aus drei URIs bestehen, werden auch als RDF-Links bezeichnet. Dabei identifizieren die URIs im Subjekt und im Objekt die verlinkten Ressourcen, während der URI, der das Prädikat identifiziert, die Art des Links angibt. Daten, die in RDF konvertiert wurden, ohne dass dabei auch RDF-Links entstanden sind, können zwar als RDF angesehen werden, sollten aber nicht als Linked Data bezeichnet werden. Erst durch die Nutzung von RDF-Links wird aus einfachem RDF Linked Data.[77]

Tim Berners-Lee schreibt: „In hypertext web sites it is considered generally rather bad etiquette not to link to related external material. The value of your own information is very much a function of what it links to, as well as the inherent value of the information within the web page. So it is also in the Semantic Web.“[78] Hintergrund ist, dass das Verlinken fremder Ressourcen die eigenen Daten in einen Kontext setzt, der dafür sorgt, dass sie möglichst gut verstanden und genutzt werden können. Ein in sich geschlossenes System entspricht nicht dem Ziel von Linked Data: Linked Data gewinnt dadurch, dass Informationen von verschiedensten Quellen zu einander in Verbindung gesetzt werden können, was nicht geht, wenn sich viele unabhängige „Dateninseln“ bilden. Informationen aus verschiedener Quelle zusammenführen zu können ist eine der besonderen Stärken von Linked Data. Bizer, Cyganiak und Heath veranschaulichen das wie folgt: „The glue that holds together the traditional document Web ist the hypertext links between HTML pages. The

[75] Vgl. Berners-Lee 2006.

[76] Bizer, Cyganiak u. a. 2007.

[77] Man kann annehmen, dass es offensichtlich ist, dass Linked Data verlinkt sein sollte. Es gibt aber Repositoriensoftware, die Daten in RDF konvertiert und dabei so gut wie keine RDF-Links setzt: Vgl. Abschnitt 3.3 ab S. 42.

[78] Berners-Lee 2006.

glue of the data web is RDF links. An RDF link simply states that one piece of data has some kind of relationship to another piece of data.“[79]

In Bezug auf Repositorien muss daher geprüft werden, welche Daten selbst bereitgestellt und welche verlinkt werden. Im Umfeld von Bibliotheken, in dem Repositorien oft angesiedelt sind, gibt es mit kontrollierten Schlagwortkatalogen und Normdaten etliche Anknüpfungspunkte. Sind diese bereits als Linked Data verfügbar, ist es leicht, entsprechende Verknüpfungen zu erstellen. Doch in Repositorien gibt es auch Metadaten, bei denen Links erst manuell erstellt oder generiert werden müssen. Ein Problem, an dessen Lösung nach wie vor gearbeitet wird, ist zum Beispiel die eindeutige Identifikation von Autoren.[80]

Da unterschiedliche Repositorien sehr heterogene Daten enthalten, reicht es nicht aus, für spezielle Metadaten, wie zum Beispiel Metadaten über die Autorenschaft eines Objektes, eine Lösung zu erarbeiten. Es muss außerdem ein generischer Mechanismus konzipiert werden, der es erlaubt, Metadaten in Links zu extern bereitgestellten Daten umzuwandeln. Alternativ kann das Datenschema eines Repositoriums erweitert werden, so dass URIs hinterlegt werden können, die bei der Umwandlung der Daten in Linked Data verwendet werden. In *Linked Data – The Story So Far* und in *How to Publish Linked Data on the Web* gibt es je einen eigenen Abschnitt dazu, wie Links manuell erstellt beziehungsweise generiert werden können.[81]

Neben der Verlinkung konkreter Objekte gibt es jedoch einen zweiten Aspekt, der maßgeblich dazu beiträgt, den Kontext der eigenen Daten über die eigene Anwendung hinaus zu verdeutlichen: Die Nutzung verbreiteter Vokabulare und Ontologien. „Different communities have specific preferences on the vocabularies they prefer to use for publishing data on the Web. The Web of Data is therefore open to arbitrary vocabularies being used in parallel. Despite this general openness, it is considered good practice to reuse terms from well-known RDF vocabularies [...] wherever possible in order to make it easier for client applications to process Linked Data.“[82] Neben der Verwendung von URIs, die in anderen Projekten für Objekte vergeben wurden, sollten auch URIs verwendet werden, die andere für Entitätstypen und Relationen vergeben haben. Es gibt einige Vokabulare und Ontologien, die oft verwendet werden. Daten, die die gleichen Vokabulare und Ontologien nutzen, können leicht zueinander ins Verhältnis gesetzt werden.

[79] Bizer, Cyganiak u. a. 2007.

[80] ORCID ist ein Projekt, das bei der Zuordnung von Forschungsergebnissen und Personen helfen soll, siehe dazu http://www.orcid.org, abgerufen am 06.06.2014.

[81] Vgl. Bizer, Heath u. a. 2009, S. 7 und Bizer, Cyganiak u. a. 2007.

[82] Bizer, Heath u. a. 2009, S. 6.

Das, was im Semantic Web als Vokabular bezeichnet wird, lässt sich gut mit dem vergleichen, was im Umfeld von Repositorien Metadatenschema genannt wird. Letztlich geht es darum die Semantik eines Wertes der veröffentlichten Daten zu spezifizieren. Genau wie bei Vokabularen gibt es auch Metadatenschemata, die weit verbreitet sind. Dublin Core ist ein gutes Beispiel, da es sowohl als Metadatenschema verwendet wird, als auch als Vokabular für Linked Data bereitsteht. Wie bereits zuvor erwähnt, wurde von der OAI festgelegt, dass alle Repositorien, die eine OAI-PMH-Schnittstelle implementieren, unter anderem Dublin Core ohne Qualifier für den Export von Metadaten über die OAI-PMH-Schnittstelle unterstützen müssen.[83]

Repositorien sind nicht auf einzelne Metadatenschemata festgelegt, sondern können in der Regel um Metadatenschemata oder zumindest um einzelne Metadatenfelder erweitert werden. Entsprechend kann hier kein Vokabular vorgeschlagen oder entwickelt werden, das so für alle Repositorien genutzt wird. Vielmehr braucht es ein entsprechend flexibles Konzept, das es ermöglicht, bei der Öffnung eines Repositoriums in Richtung des Semantic Web zu den vom Repositorium genutzten Metadatenschemata geeignete Vokabulare auszuwählen. In Kapitel 4 wird ein solches Konzept vorgestellt.

3.2 Repositoriensoftware

Es gibt viele Softwarelösungen für Repositorien. An etlichen Institutionen wurde Repositoriensoftware entwickelt, die den spezifischen Anforderungen der Institution entspricht, allerdings ausschließlich dort eingesetzt wird. Des Weiteren gibt es Softwarelösungen, die an mehreren Institutionen eingesetzt werden. In Empfehlungen zum Aufbau von Repositorien wird meist auf mehrere verschiedene Softwarelösungen hingewiesen. Dabei gibt es jedoch nur selten umfangreiche Angaben zu den verschiedenen Systemen. Aktuelle Literatur über Software für Repositorien gibt es kaum, meist wird auf einen Vergleich[84] des Open Society Institute verwiesen, dessen letzte Auflage auf August 2004 datiert ist. Des Weiteren gibt es eine Masterarbeit[85] aus dem Jahr 2008, in der Softwarelösungen für Dokumentenserver verglichen werden. Das Repositories Support Project[86] hat 2009 eine Umfrage unter Anbietern und Entwicklern von Repositoriensoftware durchgeführt und diese 2010 aktualisiert. Das Ergebnis listet in tabellarischer Form Features und Funktionen auf.[87] Die einzigen Softwarelösungen, die in allen drei Quellen genannt werden, sind DSpace,

[83] Vgl. S. 19.
[84] OSI 2004.
[85] Upmeier 2008.
[86] http://www.rsp.ac.uk, abgerufen am 06.06.2014.
[87] http://www.rsp.ac.uk/start/software-survey/results-2010/, abgerufen am 06.06.2014.

EPrints und Fedora. OSI 2004 und Upmeier 2008 berücksichtigen auch noch OPUS und MyCore.

3.2.1 Verbreitung von Repositoriensoftware

Einen Überblick über die Häufigkeit der Nutzung (Anzahl der mit einer bestimmten Software realisierten Repositorien) der Softwarelösungen bieten zwei Verzeichnisse: The Directory of Open Access Repositories (OpenDOAR)[88] und das Registry of Open Access Repositories (ROAR)[89] verzeichnen Open-Access-Repositories. Die Tabelle 3.1 gibt einen Überblick welcher Anteil der gelisteten Repositorien die jeweilige Software nutzen.

	OpenDOAR		ROAR	
DSpace	1127	41,2%	1464	39,2%
EPrints	375	13,7%	532	14,2%
OPUS	71	2,6%	54	1,4%
Fedora	41	1,5%	55	1,5%
MyCore	8	0,3%	7	0,2%
sonstige	835	30,5%	516	13,8%
unbekannt	282	10,3%	1111	29,7%
	2739		3739	

Tabelle 3.1 – Genutzte Software der in OpenDOAR und ROAR am 06.06.2014 verzeichneten Repositorien. Bei 474 der in ROAR erfassten Repositorien wird die Software mit „various“ angegeben, bei 637 Repositorien gibt es keine Angabe zur Software. Diese wurden unter unbekannt zusammengefasst. Unter sonstiger Software sind verschiedene in den Verzeichnissen benannte Softwarelösungen zusammengefasst, die jeweils von maximal 7,4% der verzeichneten Repositorien eingesetzt werden.

Die in Tabelle 3.1 angegebenen Zahlen beziehen sich auf Open-Access-Repositorien weltweit. Konzentriert man sich auf deutsche Open-Access-Repositorien, so fällt auf, dass OPUS fast ausschließlich in Deutschland verbreitet ist: 69 von 71 in OpenDOAR verzeichneten Installationen von OPUS entfallen auf Deutschland und haben dort einen Marktanteil von 41,1%. OPUS wird jedoch in Version 3 und Version 4 eingesetzt. Version 4 ist eine komplette Neuentwicklung, die nicht auf den Code

[88] http://www.opendoar.org, abgerufen am 06.06.2014.
[89] http://roar.eprints.org, abgerufen am 06.06.2014.

der Vorgängerversion aufsetzt.[90] Die Weiterentwicklung beider Versionen erfolgt getrennt: Nach der Veröffentlichung von OPUS 4.0.0 gab die Universitätsbibliothek Stuttgart bekannt, auch die Version 3 weiterzuentwickeln.[91] Die Weiterentwicklung von „OPUS 4“ wird vom Kooperativen Bibliotheksverbund Berlin-Brandenburg (KOBV) betrieben.[92] Es handelt sich daher um verschiedene Software, auch wenn OpenDOAR und ROAR diese nicht separat erfassen.[93] Der hohe Marktanteil von OPUS in Deutschland teilt sich daher auf zwei verschiedene Systeme auf.

Zusammenfassend lässt sich festhalten, dass nur DSpace und EPrints auf einen weltweiten Marktanteil von je über 10% kommen. DSpace ist gemäß den Zahlen von OpenDOAR und ROAR die mit Abstand meist genutzte Repositoriensoftware und wird von circa 40% aller Open-Access-Repositorien eingesetzt.

3.2.2 Charakteristika von Repositorien

Es gibt zwar viele verschiedene Softwarelösungen, die Ziele und die Ausrichtung gleichen sich jedoch: Von Lösungen für speziellen Repositorien abgesehen hat Repositoriensoftware die sichere Speicherung und Weitergabe digitaler Objekte und der die Objekte beschreibenden Metadaten zum Ziel.[94] Neben dem gleichen Ziel haben Bemühungen, im Bereich von Repositorien Standards, Regeln und Konventionen zu etablieren, dazu geführt, dass sich Charakteristika von Repositorien herausgebildet haben, die in den meisten Softwarelösungen berücksichtigt werden.[95] Diese Charakteristika von Repositorien sind bei der Öffnung in Richtung des Semantic Web zu beachten.

Das Reference model for an Open Archival Information System (OAIS)[96] entstand im Umfeld internationaler Raumfahrtorganisationen und beschreibt die technische und organisatorische Infrastruktur eines Langzeitarchivs. Auf ihm basiert die

[90] Der Autor dieser Arbeit hat Code zu OPUS in Version 3 beigetragen und war an der Neuentwicklung von Version 4 beteiligt.

[91] Die Ankündigung erfolgte über eine Mailingliste, deren Archiv nur für Abonnenten zugänglich ist: https://listserv.uni-stuttgart.de/mailman/private/opus-l/2011-February/000337.html, abgerufen am 06.06.2014.

[92] Vgl. http://www.kobv.de/opus4, abgerufen am 06.06.2014.

[93] Auf der Internetseite zu „OPUS 4“ listet der KOBV 41 Referenzinstallationen auf: http://www.kobv.de/opus4/referenzen/, abgerufen am 06.06.2014. Einsatzzahlen der Version 3 waren nicht zu finden.

[94] Vgl. Abschnitt 2.1 ab S. 12.

[95] Der Autor dieser Arbeit hat zur Entwicklung von DSpace und OPUS in Version 3 und 4 beigetragen und beruflich verschiedene Repositorien aufgebaut und betreut. So nicht anders angegeben, stammen die Informationen dieses Abschnitts aus persönlicher Erfahrung durch die Arbeit mit und der Begutachtung von verschiedenen Systemen.

[96] CCSDS 2012.

ISO Norm 14721:2012.[97] An Langzeitarchive gibt es weitreichendere Anforderungen als an Repositorien, dennoch hat das OAIS-Modell Einfluss auf die Entwicklung von Repositorien gehabt.[98]

Gemäß des OAIS-Modells besteht die Umgebung eines OAIS aus Produzenten, Endnutzern und dem Management. Produzenten liefern die zu erhaltenden Information, das Management legt Policies fest, und Endnutzer sind Personen oder Client-Systeme, die Informationen auffinden und erhalten wollen.[99] Als Teil eines OAIS gibt es eine Funktionseinheit „Administration“, die zum Beispiel die Übergabe von Informationen mit Produzenten vereinbart, die Übergabe prüft, Hard- und Software des Archivs betreut und so weiter.[100]

Das OAIS-Modell hat sich bei den meisten Softwarelösungen für Repositorien durchgesetzt: Ein Workflow, bei dem Objekte durch Autoren, Wissenschaftler oder andere Produzenten unter Angabe von Metadaten eingestellt, von Administratoren überprüft und anschließend für Nutzer bereitgestellt werden, ist in DSpace, EPrints und OPUS implementiert.

Das OAIS-Modell legt aufgrund seiner Ausrichtung auf Langzeitarchivierung besonderen Wert auf Metadaten, die dem Erhalt der Objekte dienen. Auch wenn die sichere und langfristige Speicherung digitaler Objekte und ihrer Metadaten Ziel von Repositorien ist, handelt es sich dabei nicht immer um Langzeitarchivierung, die nach wie vor Thema aktueller Forschung ist.[101] Im Bereich von Repositorien steht mehr die Charakterisierung der gespeicherten Objekte durch Metadaten im Vordergrund. Die Terminologie des OAIS-Modells in Bezug auf Metadaten hat sich im Repositorienumfeld bislang nicht durchgesetzt. Statt zum Beispiel von Verpackungsinformationen, Erschließungsinformationen und Erhaltungsmetadaten zu sprechen, wird dort zwischen beschreibenden, strukturellen, administrativen und technischen Metadaten differenziert.[102]

[97] Laut der deutschen Übersetzung des OAIS-Referenzmodells ist das frei verfügbare Referenzmodell inhaltlich identisch mit dem ISO-Standard, vgl. nestor 2013, S. I.

[98] Vgl. zum Beispiel eine E-Mail von Tim Donohue an eine der DSpace-Mailinglisten, in der u.a. in Bezug auf das OAIS-Modell schreibt: „DSpace is just built keeping these best practices in mind“, http://sourceforge.net/p/dspace/mailman/message/27969727/, abgerufen am 06.06.2014.

[99] CCSDS 2012, S. 2-2f.

[100] CCSDS 2012, S. 4-2.

[101] In der Langzeitarchivierung geht es um den Erhalt der Information. Dazu können digitale Objekte zum Beispiel regelmäßig konvertiert werden oder alte Systeme zur Nutzung der Objekte simuliert werden. Es gibt sowohl Repositorien, die versuchen diesem Anspruch gerecht zu werden, als auch Repositorien, die sich auf den Erhalt der digitalen Objekte auf Bitebene beschränken.

[102] Vgl. CCSDS 2012, S. 1-8ff, nestor 2013, S. 8ff, DINI 2011, S. 67 und https://wiki.duraspace.org/display/DSDOC4x/Functional+Overview#FunctionalOverview-Metadata, abgerufen am 06.06.2014.

Beschreibende Metadaten enthalten Informationen über das gespeicherte Objekt wie zum Beispiel Namen von an der Erstellung beteiligten Personen, Titel, Zusammenfassungen und andere Angaben. Strukturelle Metadaten dienen dazu, das Objekt einzuordnen und in einen Kontext zu setzen. Ein Beispiel ist die Zuordnung eines Objektes eines institutionellen Repositoriums innerhalb des Organigramms der betreibenden Institution. Auch ein Verweis von einem auf ein anderes Objekt kann als Teil der strukturellen Metadaten angesehen werden, zum Beispiel um in einem Artikel genutzte Primärdaten zu verlinken. Bei technischen Metadaten handelt es sich um Daten, die das Repositorium zur Realisierung verschiedener Funktionen benötigt und in der Regel automatisch anlegt. Das können Prüfsummen von Dateien oder auch Angaben darüber sein, wann ein Objekt ins Repositorium aufgenommen und wann es zuletzt geändert wurde. Administrative Metadaten dienen der Sicherheit und Verwaltung der Objekte eines Repositoriums. Zu ihnen gehören Benutzerkonten, Gruppen, Schreib- und Zugriffsrechte.

Repositoriensoftware extrahiert aus digitalen Objekten Text, soweit das technisch möglich ist.[103] Die extrahierten Volltexte und die Metadaten fließen in einen Suchindex ein, wobei die Metadaten in der Regel stärker gewichtet sind als die extrahierten Volltexte. Neben einer allgemeinen Suche ist eine Suche über ausgewählte Felder der Metadaten üblich. DSpace und OPUS bieten auch die Facettierung der Suchergebnisse anhand einzelner Metadatenfelder an.[104] Neben der Suche wird oft auch eine Funktion angeboten, welche die Inhalte eines Repositoriums anhand ausgewählter Metadatenfelder aufschlüsselt. Entsprechend sind beschreibende Metadaten maßgeblich für die Suche und Erschließung der Inhalte eines Repositoriums. Eine besondere Rolle spielen sie auch, wenn ein Nutzer entscheidet, ob ein bestimmtes Objekt für ihn von Interesse ist oder nicht.

Gerade im Bereich der beschreibenden Metadaten gibt es etliche Metadatenschemata, sowohl allgemeine als auch disziplinspezifische. DSpace, EPrints und OPUS bieten von sich aus eine Auswahl an Feldern für beschreibende Metadaten an. EPrints und OPUS folgen dabei keinem erkennbaren Schema, DSpace basiert auf Dublin Core gemäß dem Library Application Profile[105]. OPUS und EPrints lassen sich um einzelne Metadatenfelder erweitern, wobei das Hinzufügen eines

[103] Typisch ist eine Extraktion von Texten aus nicht durch Digital Rights Management geschützten PDF-Dateien. Eine OCR-Erkennung gehört nicht zum gängigen Funktionsumfang von Repositoriensoftware.

[104] Als Facettierung bezeichnet man eine Funktion, die verschiedene Filter auf Suchergebnisse anbietet. Dabei werden die Suchergebnisse dahingehend eingegrenzt, dass nur Objekte berücksichtigt werden, die in Bezug auf ein bestimmtes Metadatenfeld einen bestimmten Wert enthalten, also zum Beispiel von einem bestimmten Autor sind oder innerhalb eines festgelegten Zeitraums publiziert wurden.

[105] https://wiki.duraspace.org/display/DSDOC4x/Functional+Overview#FunctionalOverview-Metadata, abgerufen am 06.06.2014.

neuen Feldes bei EPrints eine Änderung des Datenbankschemas erfordert. DSpace verfolgt einen strukturierteren Ansatz, bei dem Metadatenfelder immer auch einem Metadatenschema zugeordnet sind. Metadatenschemata und zugehörige Felder lassen sich zu DSpace nach Bedarf hinzufügen, ohne dass dazu das Datenbankschema geändert werden muss.[106] Zu einem Metadatenschema speichert DSpace immer auch einen Namensraum, das heißt einen entsprechenden URI.

So gut wie alle Open-Source-Lösungen für Repositorien sind als Webapplikationen konzipiert, die relationale Datenbanksysteme zur Speicherung der Metadaten nutzen, während die digitalen Objekte selbst als herkömmliche Dateien gespeichert werden. DSpace, EPrints und OPUS stehen unter offenen Lizenzen, ihr Quellcode ist verfügbar und darf erweitert werden.[107] Sie sehen sich als „schlüsselfertige“ Applikationen für fachspezifische und institutionelle Repositorien an, die anpassbare Weboberflächen mitbringen. Eine Ausnahme stellt Fedora dar, das sich als modulare Architektur zum Aufbau von Repositorien ansieht und nicht als Lösung, die „out of the box“ eingesetzt werden kann. Die Weboberfläche eines Repositoriums teilt sich in drei Bereiche: Einen Bereich, der frei zugänglich ist und das Durchsuchen und Anzeigen der nicht zugriffsbeschränkten Objekte ermöglicht, einen Bereich für registrierte Benutzer und einen Administrationsbereich.

Für jedes Objekt eines Repositoriums gibt es eine eigene Webseite, die die Metadaten auflistet und Links auf die zugehörigen Dateien bereit hält. Im DINI-Zertifikat werden diese Seiten als„Jump-Off-Pages“ bezeichnet.[108] Es bietet sich an diese Seiten als HTML-Repräsentation zu nutzen, wie es im Abschnitt 3.1.1 ab S. 27 beschrieben ist. In der Regel gibt es auch mindestens einen Persistent Identifier, der auf die Jump-Off-Page verweist.[109]

Wie bereits dargelegt sind Persistent Identifiers im Bereich von Repositorien sehr verbreitet.[110] OPUS vergibt URNs und DSpace vergibt Handles. EPrints kann zwar Persistent Identifiers als Metadaten speichern, generiert und vergibt Persistent Identifiers jedoch nur unter Nutzung von Erweiterungen, die nicht zum offiziellen Release von EPrints gehören. Innerhalb seiner beruflichen Tätigkeit für die Technische Universität Berlin hat der Autor eine Erweiterung für DSpace

[106] Die einzige Einschränkung besteht darin, dass lediglich Metadatenschemata abgebildet werden können, die von ihrer Struktur her Dublin Core ähneln.

[107] EPrints und OPUS steht unter GPL, DSpace unter der DSpace Source Code License, eine Abwandlung der MIT-License. http://files.eprints.org/867/, http://www.kobv.de/fileadmin/opus/download/opus_dokumentation_de.pdf und http://www.dspace.org/license/, alle abgerufen am 06.06.2014.

[108] DINI 2011, S. 67.

[109] Weitere Persistent Identifiers können direkt auf einzelne Dateien des digitalen Objekts referenzieren. Vgl. hierzu auch DINI 2011, S. 64.

[110] Vgl. Abschnitt 3.1.2 ab S. 30.

entwickelt, die die parallele Vergabe von DOIs und Handles ermöglicht und seit der Version 4.0 fester Bestandteil von DSpace ist.

3.3 Repositorien und semantische Technologien

Der Begriff Semantic Web steht oft im Mittelpunkt, wenn es um semantische Technologien geht. Doch auch über das Semantic Web hinaus gibt es hilfreiche Funktionen und Konzepte, die als semantische Technologien angesehen werden. Repositorien speichern strukturiert Daten und bauen darauf Angebote für Nutzer auf. Auch wenn die Ansätze bislang noch eher schwach ausgeprägt sind, gibt es in Repositorien zunehmend Funktionen, die ausnutzen, dass die Semantik der gespeicherten Metadatenfelder festgelegt ist.

So gesehen enthalten Repositorien semantische Technologien: DSpace und OPUS bieten die Facettierung von Suchergebnissen an.[111] Mithilfe des sogenannten *browsing*[112] kann ein Überblick über die Inhalte eines Repositoriums gewonnen werden, indem die Inhalte anhand ihrer Metadaten aufgeschlüsselt werden und die Ergebnisliste durch eine Auswahl von Metadaten eingeschränkt wird. Es handelt sich quasi um die Facettierung eines Suchergebnisses, das alle Inhalte des Repositoriums umfasst. EPrints bietet keine Facettierung von Suchergebnissen an. Immerhin können bei der erweiterten Suche einzelne Metadatenfelder nach bestimmten Suchbegriffen durchsucht und mehrere solcher Suchkriterien verknüpft werden, so dass auch komplexere Suchvorgänge möglich und Suchergebnisse ähnlich wie nach der Nutzung einer Facettierung gefunden werden können.

DSpace, EPrints und OPUS ermöglichen es darüber hinaus die Metadaten über eine OAI-PMH-Schnittstelle abzurufen. DSpace und OPUS stellen die Daten jedoch nicht als Linked Data bereit, so dass sie nicht Teil des Semantic Web sind. Im Gegensatz zu DSpace und OPUS bietet EPrints einen Export der Metadaten in RDF/XML, RDF N-Triples und RDF-N3-Notation an. Dabei werden bekannte Vokabulare und Ontologien verwendet, wie zum Beispiel Dublin Core, The Bibliographic Ontology (BIBO), Friend of a Friend (foaf), SKOS und andere. Links werden nur gesetzt, wo es für die Verwendung der Ontologien unumgänglich ist, zum Beispiel um den Status eines Artikels gemäß BIBO auszuzeichnen (mittels eines Links zum Beispiel auf http://purl.org/ontology/bibo/status/nonPeerReviewed

[111] Aussagen in diesem Abschnitt zu DSpace wurden anhand der Version 4.0 nachvollzogen, Aussagen zu EPrints anhand der Version 3.3.12 und Aussagen zu OPUS anhand der Version 4.4.0.

[112] Eine Kombination der Wörter stöbern, durchsuchen und überfliegen dürfte dem englischen Begriff am nächsten kommen. Im Repositorienumfeld hat sich keine Übersetzung durchgesetzt, es wird das englische Original genutzt.

oder auf http://purl.org/ontology/bibo/status/published). Vorhandene Identifiers wie zum Beispiel ISBN und ISSN werden als urn:issn: ausgezeichnet, DOIs als info:doi:.

Eine gewöhnliche EPrints-Installation nutzt einen Teil der Library of Congress Subject Headings (LCSH) als Klassifikation. In RDF wird für jedes vergebene Schlagwort ein Link generiert, der auf das Repositorium selbst verweist. Ein Hinweis darauf, dass es sich um die bekannten LCSH handelt, fehlt gänzlich. Folgt man dem Link, erhält man eine Darstellung des von EPrints verwendeten Teils der LCSH in SKOS. Auch hier wird nicht auf die von der Library of Conress in RDF bereitgestellten LCSH verlinkt.

Auch für Autorinnen und Autoren generiert EPrints Links, die Teil des Repositoriums sind. Die von EPrints erzeugten Links enthalten Hashwerte, die auf den internen IDs der Personen sowie der jeweiligen Objekte basieren. Verfügen die Personen nicht über eine ID, werden statt dessen Vor- und Nachname sowie die ID des jeweiligen Objekts als Eingabe für den Algorithmus genommen, der den Hashwert ermittelt. Das führt in der Regel dazu, dass gleichnamige Autoren eines Objekts durch den selben URI gekennzeichnet werden, gleichnamige Autoren verschiedener Objekte durch verschiedene URIs identifiziert werden. Für einen Autor, von dem das Repositorium mehrere verschiedenen Objekte enthält, werden also mehrere URIs generiert, die nicht aufeinander hinweisen. Ein URI zur Bezeichnung einer Person lässt sich in EPrints nicht einfügen.

Abgesehen von einem Export der vorhandenen Metadaten in RDF unter Verwendung der genannten Vokabulare findet in EPrints keine Aufbereitung der im Repositorium gespeicherten Daten für das Semantic Web statt. Mögliche Verlinkungen werden nicht vorgenommen, weder auf bekannte feststehende URIs wie zum Beispiel die LCSH noch indem Möglichkeiten geschaffen werden, Links manuell anzugeben. Es werden Dateninseln erzeugt, die auf sich selbst verlinken und dabei URIs generieren, deren Erzeugung wie zum Beispiel im Fall von Autoren nicht oder nur schwer nachvollziehbar ist. Rein formal genommen handelt es sich um RDF. Von Linked Data kann aber nicht gesprochen werden, da für Linked Data grundlegende Konventionen ignoriert werden.

Kapitel 4

Einbindung von Repositorien in das Semantic Web

RDF-Daten lassen sich auf unterschiedliche Art und Weise im Internet bereitstellen. Die Auswahl einer geeigneten Methode ist von verschiedenen Kriterien abhängig: Welche Computerleistung ist erforderlich? Entstehen die bereitzustellenden Daten dynamisch, und wenn dann wie genau? Werden sie auch noch anderweitig genutzt? Welche Serialisierungen[113] der Daten sind erforderlich und sollen die Daten auch über ein SPARQL-Endpoint[114] angeboten werden?

Am einfachsten lassen sich Daten in RDF über einen Web-Server in Form statischer Dateien zur Verfügung stellen. Alternativ können RDF-Serialisierungen auch beim Abruf dynamisch generiert werden. Dazu können Wrapper[115] für relationale Datenbanken eingesetzt werden oder Programme zur Erzeugung dynamischer Webseiten erweitert werden. Triple Stores[116] dienen der Speicherung von RDF-Daten und enthalten meistens zusätzliche Funktionen zur Veröffentlichung der gespeicherten RDF-Daten im Netz. Einen guten Überblick, über gängige Methoden

[113]Für Daten, die in RDF vorliegen, gibt es verschiedene Serialisierungen. Als Serialisierung bezeichnet man ein Format zur Darstellung von Daten. Für RDF sind die Serialisierungen RDF/XML, N-Triples, die N3-Notation und Turtle gängig. Turtle ist eine Teilmenge der N3-Notation und schließt das Format N-Triples mit ein.

[114]SPARQL ist ein Protokoll und eine Anfragesprache für Daten in RDF. *SPARQL-Endpoint* ist der gängige Begriff für eine Schnittstelle, über die Daten in RDF mittels SPARQL abgerufen und gegebenenfalls auch manipuliert werden können.

[115]Als Wrapper wird eine Software bezeichnet, die eine gewünschte Schnittstelle oder Funktion mit Hilfe einer anderen vorhanden Schnittstelle oder Funktion realisiert.

[116]Als Triple Store wird eine Datenbank zur Speicherung von Daten in RDF bezeichnet. Als nativen Triple Store bezeichnet man eine solche Datenbank, wenn die Daten als RDF gespeichert und nicht erst umgewandelt werden.

RDF-Daten bereitzustellen, und über ihre Vor- und Nachteile und Hinweise auf zu beachtende Besonderheiten bietet zum Beispiel Heath und Bizer 2011.

Im Folgenden wird ein Konzept zur Bereitstellung der in Repositorien gespeicherten Metadaten als Linked Data und zur Konvertierung beziehungsweise Verlinkung der gespeicherten digitalen Objekte vorgestellt. Es orientiert sich an den Konventionen zur Veröffentlichung von Linked Data und berücksichtigt die Eigenarten von Repositorien, wie sie in den vorhergehenden Kapiteln herausgearbeitet wurden. Heath und Bizer schlagen unter anderem vor, Daten in einem Triple Store zu speichern und Tools wie zum Beispiel Pubby[117] zu nutzen, damit die genutzten URIs gemäß der Linked Data Principles dereferenzierbar werden.[118] Dieser Vorschlag wird aufgegriffen und an Repositorien angepasst: Die Repositorieninhalte werden in RDF konvertiert und in einem Triple Store gespeichert. Für die Dereferenzierung der URIs wird allerdings das Repositorium geeignet erweitert, anstatt zusätzliche Tools zu verwenden. Dieses Vorgehen bietet mehrere Vorteile: Der Triple Store kann genutzt werden, um den SPARQL-Endpoint zu realisieren, und fungiert zeitgleich als eine Art Cache für die Komponente, über die die Daten als RDF Serialisierung heruntergeladen werden können. Somit können Serialisierungen der konvertierten Daten schneller ausgeliefert werden, als wenn die Daten bei jedem Abruf dynamisch konvertiert würden. Eine Dopplung von Funktionen, die das Repositorium bereits enthält, wird vermieden. Und in den konvertierten Daten können URIs verwendet werden, die das Repositorium bereits zur Identifikation seiner Ressourcen nutzt und bei deren Dereferenzierung die Repositoriensoftware aufgerufen wird.

Das Kapitel beginnt mit der Definition von Anforderungen für die Einbindung von Repositorien in das Semantic Web. Anschließend wird ein allgemeines Konzept vorgestellt, das sich in verschiedenen Softwarelösungen für Repositorien umsetzen lässt. In Abschnitt 4.3 wird eine Proof of Concept Implementation dieses Konzepts für die Repositoriensoftware DSpace vorgestellt.[119]

[117] Pubby ist ein Tool, das Daten aus einem Triple Store gemäß der Linked Data Principles dereferenzierbar macht. Vgl. http://wifo5-03.informatik.uni-mannheim.de/pubby/, abgerufen am 06.06.2014.

[118] Vgl. Heath und Bizer 2011, S. 79.

[119] Die Proof of Concept Implementation liegt der Diplomarbeit auf einer CD bei. Nach Abschluss des Prüfungsverfahrens wird sie als Open-Source-Software mit dem Ziel einer Integration in eine offizielle Version von DSpace bereitgestellt und kann unter http://www.pnjb.de/uni/diplomarbeit/ heruntergeladen werden.

4.1 Zielsetzung

Das nachfolgende Konzept soll einige grundlegende Anforderungen erfüllen, die im Folgenden vorgestellt werden. In Kapitel 5 wird das Konzept anhand dieser Ziele evaluiert.

1. Berücksichtigung der Linked Data Principles

   Die konvertierten Daten sollen den Konventionen der Linked Data Principles[120] entsprechen. Das bedeutet insbesondere, dass die eingesetzten URIs dereferenzierbar sein sollen und mittels Content Negotiation[121] ermittelt wird, in welcher Form die Inhalte zurückgegeben werden. Die konvertierten Daten sollen in den drei gängigen Serialisierungen RDF/XML, Turtle und N-Triples bereitgestellt werden und zusätzlich über einen SPARQL-Endpoint abrufbar sein.

2. Dopplung von Funktionen vermeiden

   Eine Dopplung von Funktionen, die auch die Repositoriensoftware bereitstellt, soll vermieden werden. Zum Beispiel sollten die Jump-Off-Pages[122] der Repositorien zur HTML-Repräsentation der Metadaten verwandt werden. Die in den Repositorien bereits generierten URIs sollen beibehalten werden. Verwenden die Repositorien Persistent Identifiers[123], so sind diese in Form funktionaler URIs zu nutzen. Das heißt, dass eine DOI in der Form http://dx.doi.org/<doi> anzugeben ist und ein von der Corporation for National Research[124] (im Folgenden CNRI) vergebenes Handle in der Form http://hdl.handle.net/<handle>. Da die Konvertierung der Daten und die Erstellung der Tripel[125] konfigurierbar sein sollen (siehe unten), können Persistent Identifiers im Bedarfsfall zusätzlich als Literal mit ihrem eigenen Schema vorweg angegeben werden.

3. Berücksichtigung der softwaretechnischen Kapselung

   Die in Repositorien vorgesehene technische Kapselung soll Berücksichtigung finden. Der Zugriff auf die Repositorieninhalte soll daher über die API der Repositoriensoftware erfolgen und nicht an dieser vorbei, direkt auf die Datenbank und das Dateisystem.

[120] Vgl. Abschnitt 3.1 ab S. 26.
[121] Vgl. Abschnitt 3.1.1 ab S. 27.
[122] Vgl. S. 41.
[123] Vgl. Abschnitt 3.1.2 ab S. 30.
[124] Die Corporation for National Research Initiatives ist der Betreiber von http://hdl.handle.net. Vgl. http://www.handle.net, abgerufen am 06.06.2014.
[125] Vgl. zur Struktur von RDF S. 33.

4. Einfache Nutzung durch Repositorienbetreiber

   Eine wichtige Anforderung ist eine möglichst einfache Nutzung des Konzepts durch Repositorienbetreiber. Je einfacher das Konzept für Repositorienbetreiber umzusetzen ist, desto größer sind die Chancen, dass diese die Inhalte ihrer Repositorien als Linked Data bereitstellen. Dabei kann davon ausgegangen werden, dass die softwaretechnische Umsetzung des Konzepts nicht durch die Repositorienbetreiber erfolgt, sondern Repositorienbetreiber fertige Softwarelösungen einsetzen. Die einfache Nutzung bezieht sich somit auf die Konfiguration und den Einsatz der anhand des Konzepts erstellten Software.

5. Flexibilität und umfassende Konfigurierbarkeit

   Die Konvertierung der Metadaten in RDF muss erweiterbar sein. Nach der softwaretechnischen Umsetzung sollte die Konvertierung konfigurierbar sein. Dies ermöglicht eine Anpassung sowohl an die unterschiedlichen Metadatenschemata verschiedener Repositorien als auch der in den konvertierten Tripeln genutzten Vokabulare ohne erneute Programmierung. Dadurch soll allen Repositorienbetreibern die Möglichkeit gegeben werden, die Konvertierung der Daten so gut wie möglich an die spezifischen Inhalte ihres Repositoriums anzupassen, eine wichtige Voraussetzung für eine gute Qualität der konvertierten Daten. Im Rahmen der Konfiguration sollen Regeln definierbar sein, die Tripel abhängig vom Wert einzelner Metadatenfelder generieren und zur Erstellung von Links genutzt werden können.

6. Konvertierung oder Verlinkung der digitalen Objekte

   Ob digitale Objekte direkt in RDF konvertiert oder nur in RDF verlinkt werden können, hängt spezifisch von den digitalen Objekten ab. Viele Objekte können zum Beispiel nicht in RDF konvertiert werden, weil ihr Inhalt nur im Binärformat sinnvoll genutzt werden kann. Hinzu kommt, dass die Metadaten in Repositorien strukturiert aufgenommen werden, es für die Formate in denen die digitalen Objekte vorliegen jedoch oft keine Regeln gibt. Daher sind die Objekte der meisten Repositorien sehr heterogen. Während für die Metadaten Vorgaben erstellt werden können, wie diese abhängig von ihrer Struktur in RDF konvertiert werden, ist das für die digitalen Objekte in der Regel nicht möglich. Vielmehr muss die Entscheidung, welche digitalen Objekte umgewandelt werden und wie die Information aus den Objekten gegebenenfalls in RDF abzubilden ist, manuell für jedes einzelne Objekt getroffen werden. Es ist zu erwarten, dass die meisten Betreiber von Repositorien sich den Aufwand nicht leisten können oder wollen, alle digitalen Objekte ihres Repositoriums manuell zu bearbeiten. Das Konzept muss daher sowohl dazu in der Lage

sein die digitalen Objekte zu konvertieren, als auch auf eine Konvertierung der Objekte zu Gunsten einer Verlinkung zu verzichten.[126]

7. Erweiterbarkeit des Konzepts und seiner Umsetzung

   Eine Herausforderung im allgemeinen Umgang mit Repositorien liegt, wie auch schon bei der Definition eines Repositoriums, in der großen Heterogenität der verschiedenen Repositorien. Entsprechend kann eine Umsetzung dieses Konzepts nicht alle möglichen Repositorien abdecken und ihre Inhalte optimal als Linked Data bereitstellt. Im Konzept muss das von Anfang an berücksichtigt werden, so dass es einfach um Lösungen für spezielle Repositorien erweitert werden kann. Das betrifft sowohl die Konvertierung der Daten in RDF, als auch die Generierung von URIs. Für spezielle Repositorien sind eventuell spezielle Schritte in der Konvertierung der Daten zu berücksichtigen. Der Mechanismus zur Generierung von URIs muss austauschbar sein. Denn obwohl viele Repositorien Persistent Identifiers nutzen, kann nicht davon ausgegangen werden, dass alle Repositorien Persistent Identifiers vergeben. Optimal wäre es, wenn mehrere Mechanismen zur Bestimmung oder Generierung der in RDF verwendeten URIs parallel zur Verfügung gestellt würden und der zu verwendende Mechanismus konfigurierbar wäre.

4.2 Allgemeines Konzept

Das folgende Konzept zur Einbindung von Repositorien in das Semantic Web ist nicht auf eine bestimmte Repositoriensoftware ausgerichtet. Stattdessen kann es auf gängige Softwarelösungen für Repositorien übertragen werden.

Repositorien werden in der Regel als Webanwendungen realisiert, bei denen jeder Aufruf zur dynamischen Generierung der angeforderten Internetseite führt. Bei einigen Repositorien sollen Cachingmechanismen für einen geringeren Leistungsbedarf sorgen. Die Komponenten von Repositoriensoftware werden üblicherweise einem von drei Layern zugeordnet. Es gibt zentrale Komponenten, welche die Speicherung der digitalen Objekte und Metadaten im Dateisystem und in einem relationalen Datenbanksystem übernehmen. Diese Komponenten werden einem Storage Layer zugerechnet. Zu einem weiteren Layer zusammenfassen lassen sich die Komponenten, die die Geschäftslogik übernehmen. Der dritte Layer setzt sich aus Komponenten zur Bereitstellung von Schnittstellen und Oberflächen zusammen. Ein solches Konzept lässt sich gut mit dem Model-View-Controller Entwurfsmuster

[126] Vgl. S. 7–8.

vergleichen. Diesem entsprechend greifen die Komponenten der verschiedenen Layer aufeinander zu.[127]

Zur Umwandlung und Bereitstellung der Daten in Linked Data sollen drei Komponenten ergänzt werden: eine Komponente zur Umwandlung der Daten in RDF, ein Triple Store zur Speicherung der umgewandelten Daten und eine Komponente zum Abruf der umgewandelten Daten als Serialisierung des zugrundeliegenden RDF-Models. Abbildung 4.1 veranschaulicht schematisch den Aufbau von Repositoriensoftware und die zu ergänzenden Komponenten.

<table>
<tr><td rowspan="2">Schnittstellen / Oberflächen</td><td>Web UI</td><td>OAI-PMH-Interface</td><td>SWORD-Interface</td><td>RDF (Export)</td></tr>
<tr><td>REST</td><td>Statistik (Anzeige)</td><td>...</td><td></td></tr>
<tr><td rowspan="2">Business Logic</td><td>Rechte-management</td><td>Browse and Search</td><td>Persistend Id. Management</td><td rowspan="2">RDFUtil</td></tr>
<tr><td>Event-System</td><td>Benutzer-Verwaltung</td><td>...</td></tr>
<tr><td>Storage Layer</td><td colspan="2">Dateisystem</td><td>Relationales Datenbanksystem</td><td>Triple Store</td></tr>
</table>

Abbildung 4.1 – Beispielhafter schematischer Aufbau eines Repositoriums. Die grau unterlegten Komponenten sind bei der Umsetzung des Konzepts hinzuzufügen.

4.2.1 Triple Store und SPARQL-Endpoint

Der hinzuzufügende Triple Store dient der Aufnahme der konvertierten Daten aus dem Repositorium. Darüber hinaus stellt er die konvertierten Daten über einen – auf den lesenden Zugriff beschränkten – SPARQL-Endpoint öffentlich bereit.

Triple Stores können über eine API oder verschiedene Schnittstellen und zugehörige Protokolle genutzt werden. Etliche Anbieter bieten Triple Stores an, die

[127] Um ein Konzept zur Erweiterung von Repositoriensoftware zu erstellen, werden hier allgemeine Annahmen über den Aufbau von Repositoriensoftware vorausgesetzt. OPUS 4 und DSpace erfüllen diese Annahmen im Wesentlichen. EPrints ist genau wie OPUS und DSpace als Webanwendung realisiert und nutzt ein relationales Datenbanksystem und das Dateisystem zur Speicherung der digitalen Objekte und Metadaten. Ob der interne Aufbau von EPrints den hiesigen Angaben entspricht, wurde nicht überprüft.

sich voneinander in einer Vielzahl von Kriterien unterscheiden, zum Beispiel hinsichtlich der Lizenz, unter der sie genutzt werden können, der Schnittstellen, die sie unterstützen, oder der Menge an Daten, die sie performant verwalten können.

Der Triple Store sollte über ein offenes Protokoll an das Repositorium angebunden werden. Würde der Zugriff auf den Triple Store über eine proprietäre API erfolgen, wären Repositorienbetreiber auf den ausgewählten Triple Store festgelegt. Seit März 2013 gibt es mit SPARQL 1.1[128] eine W3C Recommendation für eine Anfragesprache zum Abruf und zur Manipulation von RDF. Die Möglichkeiten zur Manipulation von RDF waren eine der Neuerungen der Version 1.1. SPARQL 1.1 besteht aus mehreren Protokollen, unter anderem der SPARQL 1.1 Query Language[129] und dem SPARQL 1.1 Graph Store HTTP Protocol[130]. Beide sind zur Anbindung des Triple Stores geeignet, offen spezifiziert und werden von verschiedenen Triple Stores unterstützt. Durch die Verwendung von SPARQL zur Anbindung des Triple Stores ist es dem Betreiber eines Repositoriums freigestellt, welcher Triple Store zur Speicherung der konvertierten Daten genutzt wird, solange dieser SPARQL 1.1 unterstützt. Auch hinsichtlich einer Skalierung der technischen Plattform zum Betrieb eines Repositoriums ist eine Anbindung des Triple Stores über SPARQL vorteilhaft, da so bei Bedarf für den Betrieb des Triple Stores eine andere physikalische Maschine genutzt werden kann als für den Betrieb anderer Komponenten des Repositoriums.

Repositorien gliedern sich im Allgemeinen in *Items*, *Collections* und *Communities*.[131] Items repräsentieren die gespeicherten digitalen Objekte und setzen sich aus Metadaten und Dateien zusammen. Collections sind Sammlungen von Items und dienen der Gruppierung von Items und der Zuordnung von Items zu Communities. Communities können einander untergeordnet sein, so dass sich Communities in einer Baumstruktur anordnen lassen. Sie werden in der Regel genutzt, um das Organigramm einer Institution abzubilden. Neben anderen Communities enthalten Communities auch Collections, also Sammlungen von Items. Über Collections und Communities können Items in das Organigramm einer Institution eingeordnet werden. Für jedes Item, jede Collection und jede Community wird im Triple Store

[128] Vgl. http://www.w3.org/TR/2013/REC-sparql11-overview-20130321/, abgerufen am 06.06.2014.

[129] Vgl. http://www.w3.org/TR/2013/REC-sparql11-query-20130321/, abgerufen am 06.06.2014.

[130] Vgl. http://www.w3.org/TR/2013/REC-sparql11-http-rdf-update-20130321/, abgerufen am 06.06.2014.

[131] Eine entsprechende Gliederung findet sich in DSpace, EPrints und OPUS. Auch wenn es keine prinzipiellen Unterschiede gibt, werden zum Teil andere Bezeichnungen verwendet. Hier werden die Bezeichnungen von DSpace übernommen und nicht übersetzt, da auch die Klassen in DSpace entsprechend benannt sind, die auch in der Proof of Concept Implementation genutzt werden.

ein Named Graph[132] angelegt, in dem die Daten gespeichert werden, die bei der Konvertierung der jeweiligen Community, Collection oder des Items entstanden sind. Lässt sich der Triple Store so konfigurieren, dass der Default Named Graph alle Daten der übrigen Named Graphs enthält, vereinfacht das Anfragen an den SPARQL-Endpoint, die alle konvertierten Daten des Repositoriums nutzen.[133] Neben den Named Graphs für die einzelnen Bereiche des Repositoriums sollte ein Named Graph eingerichtet werden, der die Daten enthält, die das Repositorium an sich beschreiben.

Die konvertierten Daten werden vom Triple Store über einen SPARQL-Endpoint für das Semantic Web bereitgestellt. Entsprechend dürfen in den Triple Store nur Daten eingebracht werden, die keinem Zugriffsschutz unterliegen. Grund dafür ist, dass gegebenenfalls vorhandene Mechanismen des Repositoriums, die den Zugriff auf digitale Objekte und Metadaten kontrollieren, nicht für Daten im Triple Store greifen. Diese Anforderung lässt sich zur Vereinfachung der Komponente nutzen, die die konvertierten Daten als RDF-Serialisierung bereitstellt:[134] In dieser Komponente muss dann nicht mehr geprüft werden, ob ein Zugriff auf die angeforderten Daten gewährt werden darf.

4.2.2 Erweiterung des Business Layers

In der Abbildung 4.1 auf Seite 50 wird eine Komponente gekennzeichnet, die bei der Umsetzung dieses Konzepts dem Business Layer hinzugefügt werden soll und in der Abbildung als *RDFUtil* bezeichnet wird. Das RDFUtil hat keine funktionale Aufgabe, sondern ist die zentrale Schnittstelle zwischen den RDF-Komponenten untereinander sowie zwischen dem übrigen Repositorium und den RDF-Komponenten. Es dient der Bereitstellung zentraler Methoden und ruft die Komponenten auf, die die jeweilige Funktion umsetzen. Über das RDFUtil können die Konvertierung einzelner Items, Collections und Communities gestartet, Daten aus dem Triple Store gelöscht und URIs zur Identifikation der Ressourcen in RDF generiert werden. Dazu werden im Folgenden weitere Komponenten eingeführt, die vom RDFUtil angesprochen werden und daher in Abbildung 4.1 unter RDFUtil zusammengefasst wurden.

[132] Named Graphs sind eine Möglichkeit, Tripel in einem Triple Store zu bündeln. Named Graphs werden im Triple Store über einen URI adressiert mit Ausnahme des Default Named Graph. Als Default Named Graph bezeichnet man den Named Graph, innerhalb dem der Triple Store die Anfragen ausführt, die keinen Named Graph adressieren. Named Graph und Default Named Graph werden hier und im Folgenden als feste Termini nicht übersetzt.

[133] Zum Beispiel kann Fuseki, eine Software, die als Teil von Jena entwickelt wird, entsprechend konfiguriert werden.

[134] Vgl. Abschnitt 4.2.6 ab S. 58.

Den Administratoren des Repositoriums müssen Möglichkeiten geschaffen werden, die Konvertierung einzelner Bereiche oder auch aller Inhalte des Repositoriums zu starten sowie Daten aus dem Triple Store zu löschen. Dazu kann ein Command-Line-Interface erstellt oder die Weboberfläche zur Administration des Repositoriums entsprechend erweitert werden. Neben der Möglichkeit, einzelne Items, Collections und Communities umzuwandeln, wird eine Methode gebraucht, um das gesamte Repositorium zu durchlaufen und dabei die Konvertierung sämtlicher Items, Collections und Communities zu starten. Diese Methode kann genutzt werden, um vorhandene Repositorieninhalte initial im Triple Store zu speichern, zum Beispiel nach der Umsetzung dieses Konzepts in bestehenden Repositorien oder falls der Triple Store neu aufgebaut werden soll. Eine solche Methode kann in das RDFUtil oder in die Komponente integriert werden, die das Interface für die Administratoren realisiert.

Schreibende Zugriffe auf Triple Stores sollten minimiert werden, da sie sowohl zeitlich als auch von der erzeugten Rechenlast deutlich aufwändiger sind als lesende Operationen. Durch inkrementelle Updates des Triple Stores kann die Anzahl schreibender Operationen deutlich gesenkt werden. Nach einer initialen Konvertierung der vorhandenen Daten sollten daher nur im Repositorium geänderte, gelöschte oder neu aufgenommene Daten für schreibende Operationen im Triple Store sorgen und nur die jeweiligen Änderungen umsetzen.

Änderungen im Repositorium sollen automatisch im Triple Store nachvollzogen werden, möglichst direkt nach der Änderung im Repositorium. Das jeweilige Vorgehen ist abhängig von der zu erweiternden Repositoriensoftware. Zum Beispiel verfügt DSpace über einen Mechanismus, um Funktionen auszuführen, wenn Items, Collections oder Communities erzeugt, verändert oder gelöscht wurden. In Abbildung 4.1 auf Seite 50 wird dieser Mechanismus als *Event-System* bezeichnet. Er sollte genutzt werden, um die Daten im Triple Store aktuell zu halten. Wird dieses Konzept in einer Softwarelösung für Repositorien umgesetzt, die über nichts Vergleichbares verfügt, müssen Änderungen der Repositorieninhalte zeitversetzt auf andere Art und Weise im Triple Store umgesetzt werden. Repositorien speichern als Metadaten, zu welchem Zeitpunkt Daten erstellt und zuletzt aktualisiert wurden. Diese Metadaten sind für die Nutzung der konvertierten Daten sehr hilfreich und sollten daher ohnehin im Triple Store in RDF gespeichert werden. Sie können auch genutzt werden, um abzugleichen, welche Daten im Triple Store aktualisiert werden müssen. Ein solcher Prozess kann regelmäßig durch einen Cronjob[135] gestartet werden. Zur Ermittlung, welche Daten aus dem Triple Store gelöscht, im Triple

[135]Betriebssysteme enthalten in der Regel einen Dienst, über den regelmäßig auszuführende Programme (oder Scripts) gestartet werden können. Unter einigen Betriebssystemen heißt dieser Dienst cron, weswegen die auszuführenden Programme oft als Cronjobs bezeichnet werden.

Store aktualisiert oder neu in den Triple Store eingefügt werden müssen, sind dann entsprechende Methoden zu implementieren.

4.2.3 Konvertierung der Repositorieninhalte in RDF

Die Konvertierung der Daten wird durch eine Komponente gesteuert, die im folgenden als *RDFConverter* bezeichnet wird. In der Zielsetzung des Konzepts wird gefordert, dass die Konvertierung erweiterbar und hinsichtlich der Metadatenschema der zu konvertierenden Daten und der Vokabulare der konvertierten Daten flexibel ist. Erreicht werden kann das, indem der RDFConverter die Konvertierung der Daten durch Plugins vornehmen lässt, es konfigurierbar ist, welche Plugins genutzt werden, und die einzelnen Plugins selbst auch über eine Konfiguration verfügen.

Der RDFConverter übernimmt die Initialisierung der Plugins, den Aufruf der Plugins, das Zusammenfügen der von verschiedenen Plugins konvertierten Daten und einen Teil der Zugriffskontrolle. Die Zugriffskontrolle ist in den verschiedenen Softwarelösungen für Repositorien unterschiedlich stark ausgeprägt. Es gibt Repositoriensoftware, in der der Schutz einzelner Metadatenfelder, einzelner Dateien und sogar ganzer Items, Collections und Communities granular eingestellt werden kann.[136] Es sollte eine zentrale Methode geben, mit der Plugins und RDFConverter überprüfen können, ob eine Datei, ein Item, eine Collection oder eine Community im Sinne dieses Konzepts geschützt werden muss. Eine solche Methode kann zum Beispiel im RDFUtil, der Schnittstelle zwischen RDF-Komponente und Repositorium, angesiedelt werden, wenn es eine solche Funktion nicht ohnehin in der Repositoriensoftware gibt. Wie in Abschnitt 4.2.1 ausgeführt, sollen nur Daten konvertiert und im Triple Store gespeichert werden, die keiner Zugriffsbeschränkung unterliegen. Der RDFConverter startet die Konvertierung nur für solche Items, Collections und Communities. Plugins dürfen andere Dateien, Items, Collections und/oder Communities nur verlinken, wenn sie öffentlich bereitgestellt werden. Bevor Plugins bei der Konvertierung den Wert eines Metadatenfeldes nutzen, müssen sie zudem prüfen, ob das Metadatenfeld einem Schutz unterliegt.

Jedes Plugin enthält eine Methode, über die sich bestimmen lässt, ob es die Konvertierung des jeweiligen Typs (Item, Collection, Community oder Beschreibung des Repositoriums) unterstützt. Der RDFConverter ruft zu einem zu konvertierenden Objekt nur die Plugins auf, die den jeweiligen Typ unterstützen.

[136] Zum Beispiel lassen sich in DSpace Metadatenfelder konfigurieren, die nur für Administratoren einsehbar sein sollen. Darüber hinaus lassen sich die Rechte von anonymen und angemeldeten Nutzern sowie von Benutzergruppen für jede Datei, jedes Item, jede Collection und jede Community einstellen.

4.2.4 Plugins zur Konvertierung

Durch den Einsatz von Plugins zur Konvertierung der Repositorieninhalte in RDF, können prinzipiell beliebige Konvertierungen umgesetzt werden. Es muss mindestens ein Plugin zur Konvertierung der gespeicherten Metadaten entwickelt werden. Des Weiteren braucht es eine Abbildung der Beziehungen zwischen Items, Collections und Communities in RDF. Zusätzlich soll auch eine Beschreibung des Repositoriums an sich in RDF erfolgen. Auch diese Daten werden von Plugins erzeugt. Werden die Communities, die keiner anderen Community zugehörig sind (in DSpace als *TOP-Level-Community* bezeichnet) in den Daten zur Beschreibung des Repositoriums referenziert, kann diese Beschreibung als zentraler Einstiegspunkt genutzt werden. Dadurch wird eine Möglichkeit geschaffen, von RDF-Link zu RDF-Link das gesamte Repositorium zu durchwandern. Auch für den umgekehrten Weg sollten die entsprechenden Links generiert werden, das heißt Links, die von Items auf ihre Collections verweisen, von Collections auf Communities und so weiter, bis hin zu den Daten, die das Repositorium beschreiben.

Die in einem Repositorium gespeicherten Dateien stehen immer in direktem Zusammenhang mit den Items, denen sie zugeordnet sind. Ein Item kann aus mehreren Dateien bestehen. Jede Datei ist genau einem Item zugeordnet. Der RDF-Converter ruft keine Plugins zur Konvertierung einzelner Dateien auf. Ein Plugin kann alle oder einzelne Dateien eines Items verlinken oder in RDF konvertieren, wenn der RDFConverter das Plugin startet, um das Item zu konvertieren, dem die Dateien zugeordnet sind. Neben dem direkten Zusammenhang zwischen Dateien und den Items, denen sie zugeordnet sind, spielt dabei auch eine Rolle, dass so alle Dateien eines Items einem Plugin gemeinsam zur Konvertierung bereit stehen und nicht nur einzelne Dateien. Ein Plugin kann dadurch entscheiden, ob es mehrere Dateien gemeinsam oder voneinander gesondert behandelt, was vom Item und den jeweiligen Dateien abhängig sein kann. Durch das Schreiben entsprechender Plugins können gegebenenfalls im Repositorium gespeicherte Dateien in RDF umgesetzt werden, anstatt nur verlinkt zu werden.[137] Das Verlinken von Dateien kann auch in das Plugin integriert werden, das die Beziehungen zwischen Items, Collections und Communities in RDF abbildet.

Die Plugins sollten konfigurierbar sein, insbesondere das oder die Plugins zur Konvertierung der im Repositorium gespeicherten Metadaten. Im Rahmen der Konfiguration müssen sich Regeln ausdrücken lassen, die definieren aus welchen Metadatenfeldern welche Tripel generiert werden. Neben der Angabe des Namens eines Metadatenfeldes muss es möglich sein, Bedingungen für den Wert des Metadatenfel-

[137] Vgl. bezüglich der Frage, ob Dateien verlinkt oder konvertiert werden sollen, den entsprechenden Abschnitt der Zielsetzung ab S. 48 und die Diskussion auf S. 7–8.

des anzugeben, so dass Tripel nicht nur abhängig vom Namen des Metadatenfeldes sondern auch von seinem Wert erzeugt werden können. Es muss möglich sein, den Wert eines Metadatenfeldes bei der Erzeugung der Tripel zu verwenden und dabei auch zu verändern. Der gegebenenfalls angepasste Wert eines Metadatenfeldes sollte sowohl in Literalen als auch in URIs genutzt werden können. Umsetzen lässt sich das zum Beispiel, indem die Konfiguration die Angabe regulärer Ausdrücke vorsieht, die einen Filter auf den Wert eines Metadatenfeldes definieren oder spezifizieren, wie ein Subjekt, Prädikat oder Objekt aus dem Wert eines Feldes gebildet werden soll. Prinzipiell kann eine solche Konfiguration in RDF angegeben werden. Das hat den Vorteil, dass durch den Einsatz von Reification[138] die zu erzeugenden Tripel in RDF angegeben werden können und so der volle Umfang von RDF genutzt werden kann, wie zum Beispiel die Typisierung von oder Sprachangaben zu Literalen. Zur Umsetzung der Konfiguration in RDF ist ein entsprechendes Vokabular zu entwickeln. Ein solches Vokabular ist stark von den Aufgaben und dem Funktionsumfang des zu konfigurierenden Plugins abhängig. Im Rahmen der Implementation wurde ein solches Vokabular entwickelt, das als Beispiel dienen kann.[139]

Bei der Konvertierung der Metadaten ist generell darauf zu achten, nur vorhandene Informationen auszugeben und Informationen nicht nach Belieben zu generieren. Gibt es bereits URIs, die zur Identifikation in RDF verwendet werden können, sollten keine neuen URIs für den selben Zweck generiert, sondern vorhandene URIs genutzt werden. Für die Generierung neuer URIs müssen außerdem genügen Informationen vorhanden sein. Die in Repositorien gespeicherten Metadaten reichen oft nicht aus, um eine Ressource eindeutig zu identifizieren. Zum Beispiel speichern Repositorien von Autoren in der Regel nur den Vor- und Nachnamen sowie gegebenenfalls den Titel. Sollte nun für jeden Autoren eines Repositoriums ein URI generiert werden, müsste entweder für jedes Dokument eines Autors ein neuer URI für den Autor generiert werden, oder alle Autoren mit dem selben Namen würden als ein einziger Autor geführt werden. Beides wäre nicht richtig.[140] Darüber hinaus pflegt ein Autor selbst vielleicht schon einen URI, der zum Beispiel auf ein foaf-Profil[141] des Autors weiterleitet. Allein aus dem Namen eines Autors kann

[138] *Reification* bezeichnet den Prozess, Aussagen über Aussagen zu treffen, die auch als *reified statement* bezeichnet werden. In der Konfiguration eines Plugins kann Reification genutzt werden, um die Tripel anzugeben, die bei der Konvertierung von Metadaten erzeugt werden sollen.

[139] Das Plugin zur Konvertierung von Metadaten, das im Rahmen der Implementation entwickelt wurde, und das Vokabular zu seiner Konfiguration werden im Abschnitt 4.3.6 ab S. 69 näher dargestellt.

[140] In Repositorien, die Autoren eindeutig identifizieren, treten diese Probleme nicht auf. Für die meisten Repositorien ist die Autorenidentifikation jedoch noch nicht gelöst. Ein Projekt, das aktuell daran arbeitet, Probleme im Rahmen der Autorenidentifikation zu lösen, ist zum Beispiel ORCID. Siehe dazu http://www.orcid.org, abgerufen am 06.06.2014.

[141] Friend of a Friend (foaf) ist ein bekanntes Vokabular, mit dem Angaben zu einer Person in RDF gemacht werden können. Ziel ist es maschineninterpretierbar Informationen bereitzustellen,

dieser URI nicht hergeleitet werden. Selbst wenn in einem Repositorium Autoren eindeutig erkannt werden könnten, würde ein vom Repositorium generierter URI für den Autor nicht dem URI entsprechen, der sonst zur Identifikation des Autors genutzt wird.[142] In solchen Fällen ist die Angabe des Autorennamens als Literal gegenüber der Generierung eines URIs zu bevorzugen. Ein anderes Beispiel sind Klassifikationen, für die es bereits feste URIs gibt. Ein Metadatenfeld, dass zum Beispiel den Wert eines Items gemäß der Library of Congress Subject Headings (LCSH) enthält, kann durch den von der Library of Congress vergebenen URI abhängig vom Wert des Metadatenfeldes umgesetzt werden. Das lässt sich mit der beschriebenen Konfiguration einfach umsetzen und jederzeit anpassen, ohne Programmcode ändern zu müssen. Bei Bedarf könnte auch ein Plugin entwickelt werden, das den Wert eines Metadatenfeldes an einen Webservice übergibt, der einen geeigneten URI zurückliefert, der in den konvertierten Daten zu nutzen ist.

Die Metadaten in Repositorien sind grundsätzlich von hoher Qualität, da sie manuell erzeugt und in der Regel zusätzlich überprüft wurden. Bei der Umwandlung in Linked Data sollte jedoch behutsam vorgegangen werden, um die Qualität der Metadaten nicht zum Beispiel durch unnötig erzeugte URIs oder falsch gesetzte Links zu verschlechtern. Dieses Konzept kann nur die technischen Grundlagen für die Konvertierung in Linked Data liefern. Entsprechend stark hängt die Qualität der Konvertierung von der Konfiguration ab.

4.2.5 URIs für Ressourcen des Repositoriums

Zur Identifikation der Ressourcen des Repositoriums (Dateien, Items, Collections, Communities) in RDF müssen URIs generiert werden, die in RDF genutzt werden können. In Kapitel 3 wurde die Nutzung von URIs in Repositorien ausführlich diskutiert, insbesondere die Abschnitte 3.1.1 und 3.1.2 ab S. 27 gehen auf die verschiedenen URIs und Persistent Identifiers ein. Demnach sollen die URIs, die das Repositorium bereits vergeben hat, zur Identifikation in RDF genutzt werden und gegebenenfalls Persistent Identifiers in Form funktionaler URIs Verwendung finden, das heißt als HTTP- oder HTTPS-URIs. Ausschließlich URIs zum Abruf der verschiedenen Serialisierungen von RDF sollen generiert werden.

wie sie sonst oft in sozialen Netzwerken veröffentlicht werden, weshalb von einem foaf-Profil gesprochen werden kann. Vgl. http://www.foaf-project.org, abgerufen am 06.06.2014.

[142] Mittels owl:sameAs kann gekennzeichnet werden, dass zwei URIs dieselbe Ressource identifizieren. Verschiedene URIs für dieselbe Ressource stellen insofern kein prinzipielles Problem dar. Würde jedoch jedes Linked Data Projekt eigene URIs für Ressourcen generieren, für die es bereits einen oder mehrere URIs gibt, würde das Zusammenführen von Daten immer aufwändiger.

Welche URIs in einem Repositorium genutzt werden, hängt sowohl von der Repositoriensoftware als auch von der Konfiguration ab und kann von Repositorium zu Repositorium verschieden sein. Repositorien, die die Software OPUS nutzen, vergeben in der Regel URNs und geben diese auch als Referenzadresse an, das heißt als Adresse, die in Zitationen genutzt werden soll. Repositorien, die auf DSpace aufsetzen, nutzen zur Adressierung Handles. Allerdings ist es dem Repositorienbetreiber freigestellt, ob ein CNRI-Prefix[143] zur Handlevergabe genutzt wird oder auf die Nutzung von hdl.handle.net zur Dereferenzierung von Handles verzichtet wird. In DSpace lässt sich konfigurieren, durch welchen Präfix Handles in HTTP-URIs umgewandelt werden, so dass die Referenzadresse auch unterhalb der Domain gebildet werden kann, unter der das Repositorium läuft. Seit DSpace 4.0 gibt es darüber hinaus die Möglichkeit parallel dazu DOIs zu registrieren und diese zu verwenden. EPrints nutzt in der Regel URIs aus dem Adressraum des Repositoriums, das heißt unterhalb der Domain über die das Repositorium adressiert wird.

Da die genutzten URIs nicht nur von der Repositoriensoftware, sondern auch von der Konfiguration abhängen, müssen alle in Frage kommenden Algorithmen zur Generierung von URIs implementiert werden. Ziel ist eine Methode, die zu einer Datei, einem Item, einer Collection oder Community den URI zur Identifikation zurück liefert. Per Konfiguration muss auswählbar sein, welcher Algorithmus letztlich genutzt wird, da in der Regel nicht automatisch erkannt werden kann, welche URIs auf den Jump-Off-Pages[144] zur Zitation angezeigt werden.

Für die RDF-Serialisierungen sollten URIs gewählt werden, deren Form den Adressen der Jump-Off-Pages ähnelt. Die genaue Form dieser URIs ist letztlich abhängig davon, wie die Komponente adressiert werden kann, die die konvertierten Daten als Serialisierung von RDF ausgibt.

4.2.6 Bereitstellen der Daten als RDF-Serialisierung

Im Rahmen der Linked Data Principles weist Tim Berners-Lee daraufhin, dass RDF Daten nicht nur über einen SPARQL-Endpoint, sondern immer auch in einer Serialisierung von RDF bereitgestellt werden sollten.[145] Die meisten Repositorien stellen ihre Daten als HTML bereit, indem sie die Daten aus der Datenbank laden und die entsprechende HTML-Darstellung dynamisch generieren zu dem Zeitpunkt, in dem eine entsprechende URL dereferenziert wird. Gegebenenfalls sorgen Cachingmechanismen für eine Entlastung der oder des Web-, Application- und Datenbankserver(s). Für die Bereitstellung der Serialisierung von RDF

[143] Vgl. Fußnote 124 auf Seite 47.
[144] Vgl. S. 41.
[145] Vgl. S. 33.

kann eine vergleichbare Strategie genutzt werden. Dazu wird eine Webanwendng implementiert, die die konvertierten Daten in RDF aus dem Triple Store lädt und als RDF/XML, Turtle und N-Triples bereitstellt. Neben der Bereitstellung eines SPARQL-Endpoints sorgt die Speicherung der konvertierten Daten in einem Triple Store dafür, dass Repositorieninhalte nicht erst zum Zeitpunkt des Abrufs konvertiert werden müssen. Der Triple Store wirkt dadurch wie ein Cache, in dem die konvertierten Daten vorgehalten werden.

Die URIs, die zur Identifikation der Ressourcen in RDF verwendet werden, sollen gemäß den Linked Data Principles dereferenzierbar sein. Dieses Konzept sieht vor, die URIs zu verwenden, die das Repositorium zur Referenzierung des jeweiligen Items, der Collection oder Community angibt. Das bedeutet jedoch, dass diese URIs bereits auf die HTML-Repräsentation des jeweiligen Items, der Collection oder Community weiterleiten. Entsprechend muss das Repositorium um Content Negotiation erweitert werden, damit beim Aufruf eines entsprechenden URIs gegebenenfalls zu der Komponente umgeleitet werden kann, die die Daten in einer RDF-Serialisierung bereitstellt. Bei der Content Negotiation sollte im Zweifelsfall die Ausgabe von HTML bevorzugt werden, wenn die URIs bislang bereits HTML anzeigten. Das kann zum Beispiel erforderlich sein, wenn vom Client kein HTTP-Accept-Header[146] gesendet wird oder es sich um einen Browser handelt, der Probleme in Zusammenhang mit Content Negotiation verursacht.[147] Damit soll gewährleistet werden, dass ältere Clients, zum Beispiel Browser, die Probleme mit Content Negotiation verursachen, oder andere Programme weiterhin auf die Inhalte zugreifen können, die bislang unter den entsprechenden URIs zu finden waren.

4.3 Proof of Concept Implementation

Zur Überprüfung des Konzepts wurde im Rahmen der vorliegenden Arbeit eine Proof of Concept Implementation erstellt, die auf DSpace[148] aufsetzt, der meist verbreitetsten Repositoriensoftware.[149] DSpace ist hauptsächlich in Java geschrieben und nutzt eine Postgresql- oder eine Oracle-Datenbank zur Speicherung der

[146]Der HTTP-Accept-Header wird in HTTP von Clients genutzt, um anzugeben, in welchen Formaten eine Antwort zurück gesandt werden soll. Dabei ist es möglich, eine Priorität zwischen verschiedenen Formaten anzugeben und auch Formate gänzlich auszuschließen.

[147]Zum Beispiel sendet der Internet Explorer bis einschließlich zur Version 8 einen HTTP-Accept-Header, der jeglichen Mime-Type akzeptiert, so dass keine zuverlässige Content Negotiation durchgeführt werden kann.

[148]Die folgenden Informationen zu DSpace entstammen der offiziellen Dokumentation, die unter https://wiki.duraspace.org/display/DSDOC zu finden ist (abgerufen am 06.06.2014) und der Erfahrung des Autors in der Arbeit mit und der Weiterentwicklung von DSpace.

[149]Vgl. Abschnitt 3.2.1 ab S. 37.

Metadaten. Die im Repositorium gespeicherten Dateien werden in der Regel im Dateisystem abgelegt. Alternativ kann eine Speicherinfrastruktur namens Storage Resource Broker[150] eingesetzt werden. Für den Betrieb wird ein Servlet Container benötigt, zum Beispiel Tomcat[151] oder Jetty[152]. DSpace setzt Apache Maven[153] (im Folgenden Maven) zum Kompilieren und für das Dependency-Management ein. Die Installation erfolgt mittels Ant.[154] DSpace wird unter der DSpace Source Code License[155] veröffentlicht, die an die MIT-License angelehnt ist, der Quellcode wird über GitHub[156] bereitgestellt.

4.3.1 Erweiterung von DSpace

DSpace setzt sich aus mehreren Modulen zusammen, die teilweise voneinander abhängig sind, teilweise unabhängig von einander verwendet werden können. Den Kern bilden die Module `dspace-api` und `dspace-services`, die Grundfunktionen von DSpace bereitstellen. Das Modul `dspace-solr` realisiert Browsing- und Suchfunktionalitäten unter Nutzung des Frameworks Apache Solr[157]. Es gibt zwei Module, die eine Weboberfläche anbieten: `dspace-jspui` und `dspace-xmlui`. In einzelnen Versionen von DSpace gibt es Funktionen, die nur von einer der beiden Oberflächen unterstützt werden. In der Regel werden solche Funktionen in einer späteren Version von DSpace auch in die jeweils andere Oberfläche portiert.[158] Im Wesentlichen unterscheiden sich beide Oberflächen in Bezug auf die eingesetzten Technologien: `dspace-jspui` setzt auf Java Server Pages auf, `dspace-xmlui` nutzt das Framework Cocoon[159] und XSL-Transformationen zur Generierung der Oberfläche. Die Module `dspace-lni`, `dspace-oai`, `dspace-rest`, `dspace-sword` und `dspace-sword2` bieten verschiedene Schnittstellen für DSpace an. Alle genannten Module, mit Ausnahme von `dspace-api` und `dspace-services`, erzeugen beim Kompilieren eine Webanwendung, die im Servlet-Container deployed[160] werden

[150] Vgl. http://www.sdsc.edu/srb/index.php/Main_Page, abgerufen am 06.06.2014.
[151] Vgl. http://tomcat.apache.org/, abgerufen am 06.06.2014.
[152] Vgl. http://www.eclipse.org/jetty/, abgerufen am 06.06.2014.
[153] Vgl. http://maven.apache.org, abgerufen am 06.06.2014.
[154] Früher wurde Ant (http://ant.apache.org, abgerufen am 06.06.2014) zur Kompilation und Installation genutzt. Einige Prozesse wurden noch nicht von Ant auf Maven portiert, so dass derzeit beide Systeme zum Einsatz kommen.
[155] Vgl. http://www.dspace.org/license/, abgerufen am 06.06.2014.
[156] Vgl. https://github.com/DSpace/DSpace, abgerufen am 06.06.2014.
[157] Vgl. http://lucene.apache.org/solr/, abgerufen am 07.06.2014.
[158] Ein Beispiel ist die Funktion zur Versionierung von Items, die mit DSpace 3.0 nur für `dspace-xmlui` verfügbar war, und mit der Version 4.0 auch in `dspace-jspui` Einzug hielt.
[159] Vgl. http://cocoon.apache.org, abgerufen am 06.06.2014.
[160] Als *Deployment* wird der Bereitstellungsprozess von Webanwendungen in Java Servlet Containern bezeichnet, also das Entpacken, Laden, Konfigurieren und Starten von Webanwendungen

kann. Dabei ist `dspace-solr` die einzige Webanwendung, die von einigen der anderen Webanwendungen vorausgesetzt wird. In einem weiteren repository[161] auf GitHub wird außerdem das Modul `dspace-lang` bereitgehalten, das via Maven als Dependency während des Build-Prozesses geladen wird und Sprachangaben für die Internationalisierung der Oberflächen enthält.

Die Implementation wurde als ein eigenes Modul namens `dspace-rdf` umgesetzt. Das Modul erstellt eine Webanwendung und eine Programmbibliothek. Die Programmbibliothek enthält alle Klassen des Moduls, die Webanwendung im Wesentlichen das `DataProviderServlet`. Die Webanwendung kann in einem Servlet-Container deployed werden. Sie lädt die konvertierten Daten aus dem Triple Store und liefert sie als RDF-Serialisierung in der Form RDF/XML, Turtle oder N-Triples aus.

Einige Klassen von DSpace sind für diese Implementation von Interesse und werden in der API von `dspace-rdf` genutzt. Die Klasse `DSpaceObject` aus dem Paket `org.dspace.content` ist eine Oberklasse der wichtigsten Entitäten von DSpace, wie zum Beispiel der Klassen `Item`, `Collection` und `Community`. Im Paket `org.dspace.core` ist die Klasse `Context` enthalten, in der der Kontext einer Operation abgebildet wird. Sie enthält zum Beispiel eine initialisierte Datenbankverbindung oder die Information, ob die aktuelle Operation mit den Rechten eines bestimmten und gegebenenfalls welchen Nutzers ausgeführt wird. Entsprechend wird eine Instanz dieser Klasse für Operationen benötigt, die auf die Datenbank zugreifen oder Operationen ausführen, für die eine Berechtigung erforderlich ist. Zur Verarbeitung von RDF und zur Nutzung von SPARQL wird die freie und offene Programmbibliothek Apache Jena[162] (im Folgenden Jena) genutzt. Jena steht unter einer Lizenz, die zur DSpace Source Code License kompatibel ist. Jena hält Daten in RDF in einem sogenannten `Model`. Das `Model` ist ein Interface, das von Klassen implementiert wird, die die Daten im Arbeitsspeicher halten oder im Dateisystem speichern. Darüber hinaus gibt es die Interfaces `Statement`, `Resource`, `Property` und `Literal`, die die entsprechenden Einheiten von RDF in Jena repräsentieren.

durch den Servlet Container. Das englische Verb *deploy* wird hier und im Folgenden als fester Terminus genutzt.

[161] Zur Abgrenzung gegenüber der Art von Repositorien, die im Zentrum der vorliegenden Arbeit stehen, wird hier das englische Wort repository verwendet. Dabei ist der Speicherort eines Versionsverwaltungssystems gemeint, in diesem Fall des Versionsverwaltungssystems Git.

[162] Vgl. http://jena.apache.org, abgerufen am 06.06.2014.

4.3.2 Aufbau von dspace-rdf

Ein Überblick über `dspace-rdf` bietet das vereinfachte Klassendiagramm in Abbildung 4.2 auf der nächsten Seite. Das Modul wird durch das Paket `org.dspace.rdf` umgesetzt. Dieses enthält vier zentrale Klassen und die drei Pakete `org.dspace.rdf.storage`, `org.dspace.rdf.conversion` und `org.dspace.rdf.providing`. Das Paket `org.dspace.rdf.storage` bündelt alle Klassen für die Speicherung und Identifikation der Ressourcen des Repositoriums in RDF. Es enthält somit alle Klassen zur Anbindung des Triple Stores an das Repositorium. `org.dspace.rdf.conversion` enthält die Klassen zur Konvertierung der Daten in RDF. `org.dspace.rdf.providing` besteht aus dem `DataProviderServlet` zur Bereitstellung der Daten als Serialisierung von RDF und dem Paket `org.dspace.rdf.providing.negotiation` für die Content Negotiation.

Die APIs der Komponenten zur Anbindung des Triple Stores, zur Generierung von URIs und zur Konvertierung der Daten wurden in Interfaces festgeschrieben. Dadurch können die einzelnen Klassen einfach ausgetauscht werden, falls andere Protokolle oder Algorithmen zum Einsatz kommen sollen. Die Klassen der einzelnen Pakete werden in den folgenden Abschnitten detaillierter vorgestellt: `org.dspace.storage` in den Abschnitten 4.3.3 und 4.3.4, `org.dspace.conversion` in den Abschnitten 4.3.5 und 4.3.6, `org.dspace.providing` im Abschnitt 4.3.7.

Die vier zentralen Klassen des Pakets `org.dspace.rdf` heißen `RDFConsumer`, `RDFUtil`, `RDFizer` und `RDFConfiguration`. Der `RDFConsumer` bindet `dspace-rdf` in das Event-System von DSpace ein. Er sorgt also dafür, dass Änderungen im Repositorium im Triple Store nachvollzogen werden. Das `RDFUtil` ist die zentrale Schnittstelle zwischen den RDF-Komponenten untereinander sowie zwischen dem übrigen Repositorium und den RDF-Komponenten. Die Klasse `RDFizer` stellt das in Abschnitt 4.2.2 geforderte Interface für Administratoren bereit, das wie in DSpace üblich über die Kommandozeile genutzt wird. Diese drei Klassen setzen die in Abschnitt 4.2.2 konzipierten Funktionen um.

Die Klasse `RDFConfiguration` lädt die Konfiguration und stellt verschiedene Methoden zur Nutzung der Konfiguration für andere Klassen des Moduls `dspace-rdf` bereit. Für die meisten Komponenten, für die in der Konfiguration festgelegt werden kann, welche Klassen verwandt werden sollen, initialisiert die `RDFConfiguration` die entsprechenden Objekte. In DSpace wird an einigen Stellen das Spring Framework[163] genutzt, um zu konfigurieren, welche Komponenten zum Einsatz kommen. An anderen Stellen findet eine Konfiguration der zu nutzenden Komponenten in den Konfigurationsdateien statt, die DSpace direkt einliest, bevor

[163] Vgl. http://spring.io, abgerufen am 06.06.2014.

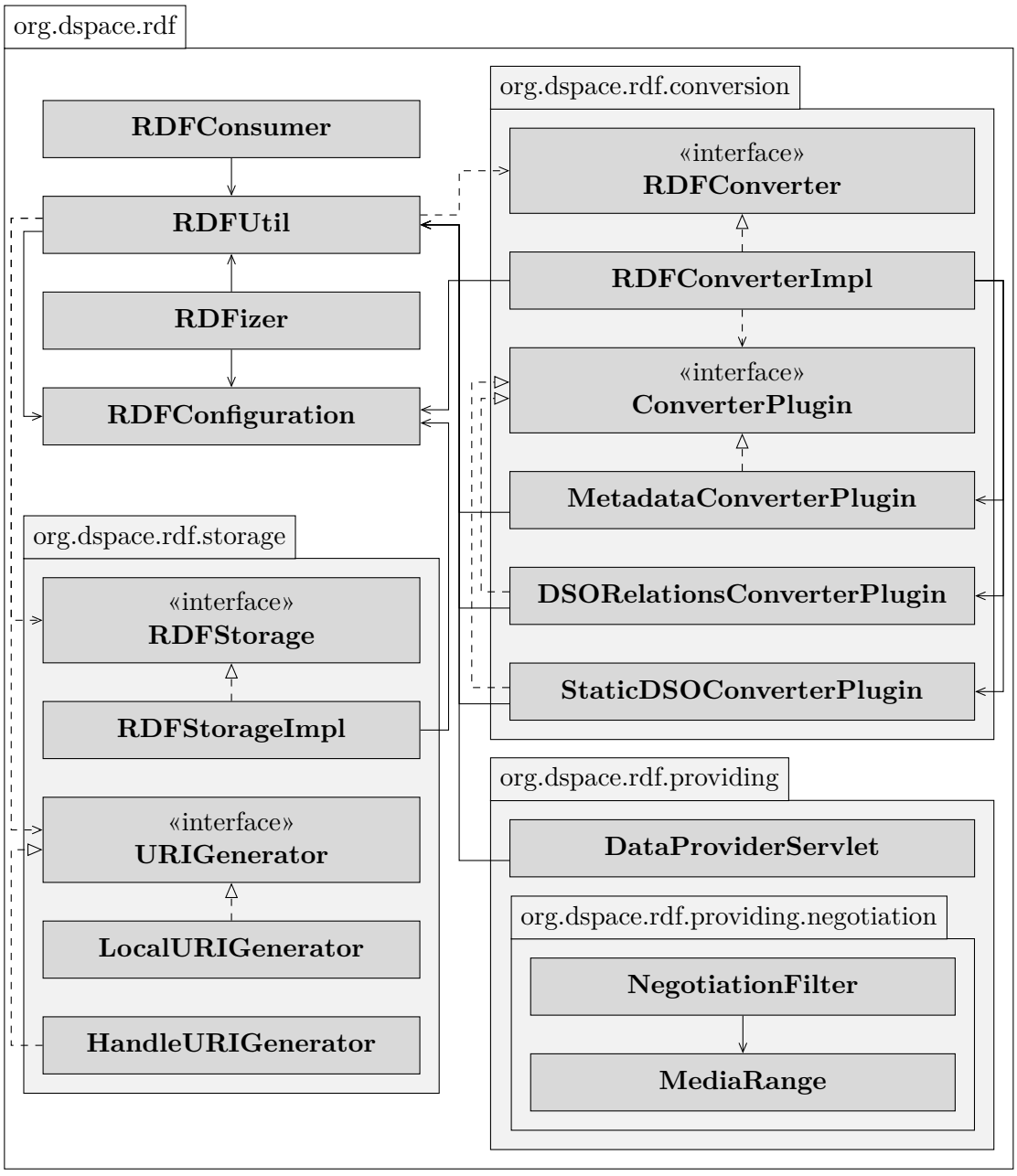

Abbildung 4.2 – Vereinfachtes Klassendiagramm der Proof of Concept Implementation.

es die Komponenten selbst initialisiert. Für diese Implementation wurde bewusst auf Spring verzichtet, um die Konfiguration für die späteren Nutzer möglichst einfach zu halten. DSpace nutzt zur Konfiguration von Spring XML-Dateien, die für Endnutzer nicht so einfach anzupassen sind wie die übrigen DSpace Konfigurationsdateien. Zudem ist es einfacher die Konfiguration sinnvoll zu gliedern, wenn auf Spring verzichtet wird. Zum Laden der Konfiguration und zur Initialisierung der erforderlichen Klassen wurde deshalb die Klasse `RDFConfiguration` entwickelt.

Bei der Entwicklung dieser Implementation wurden zwei Probleme im DSpace-Event-System entdeckt, die Auswirkungen auf den `RDFConsumer` und die Nutzung von DOIs als URIs in RDF haben. Zum einen wurden keine Events erstellt, wenn Top-Level-Communities[164] gelöscht wurden. Dieses Problem konnte durch den Autor dieser Arbeit behoben werden, der entsprechende Bugfix wurde in den offiziellen Quellcode von DSpace bereits aufgenommen.[165]

Das andere Problem im Event-System betrifft das Löschen von Objekten im Triple Store, wenn die korrespondierenden Objekte in DSpace gelöscht wurden. Das Event-System löst Events erst aus, nachdem die zugrundeliegenden Änderungen in die Datenbank geschrieben wurden. Wird eine Ressource des Repositoriums in DSpace gelöscht, wird das Event erst ausgelöst, wenn die jeweiligen Daten tatsächlich aus der Datenbank entfernt wurden. Das entsprechende `DSpaceObject` kann dann nicht mehr initialisiert werden, da die Daten nicht mehr in der Datenbank gespeichert sind. Die Events enthalten den Typ, die interne ID und gegebenenfalls das Handle eines Objekts, jedoch nicht die übrigen Metadaten. Diese Angaben reichen nicht, um zum Beispiel zu ermitteln, ob einem gelöschten Objekt eine DOI zugeordnet war und gegebenenfalls wie diese DOI lautete. Würden DOIs in `dspace-rdf` zur Identifikation von Items, Collections oder Communities genutzt, könnten Löschungen im Repositorium nicht im Triple Store nachvollzogen werden, da der zu löschende Named Graph nicht oder nur sehr aufwändig ermittelt werden könnte. Eine mögliche Lösung des Problems, bei der ein Event alle Persistent Identifiers eines betroffenen `DSpaceObject`s enthält, wurde im Rahmen dieser Implementation entwickelt. Bevor sie in den Quellcode von DSpace übernommen wird, sind jedoch umfassende Tests erforderlich.[166]

4.3.3 Anbindung des Triple Stores

Das Interface `RDFStorage` schreibt die Implementierung von Funktionen zum Speichern, Laden und Löschen von Daten im Triple Store vor und ist in Abbildung

[164] Vgl. Abschnitt 4.2.4 ab S. 55.
[165] Vgl. https://jira.duraspace.org/browse/DS-1966, abgerufen am 06.06.2014.
[166] Vgl. https://jira.duraspace.org/browse/DS-1990, abgerufen am 06.06.2014.

4.3 zu sehen. Des Weiteren enthält es eine Funktion, die die URIs aller im Triple Store enthaltenen Named Graphs als Liste zurückliefert. Die Daten, die im Triple Store gespeichert werden sollen oder aus dem Triple Store geladen wurden, werden in der Datenstruktur `Model` der Programmbibliothek Jena übergeben.

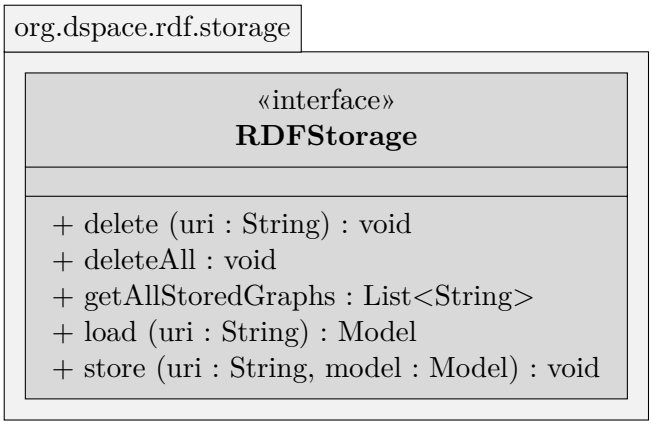

Abbildung 4.3 – Klassendiagramm des Interfaces RDFStorage.

Zur Anbindung des Triple Stores wurde die Klasse `RDFStorageImpl` entwickelt, die sowohl das SPARQL 1.1 Graph Store HTTP Protocol als auch die SPARQL 1.1 Query Language nutzt. Hintergrund ist, dass es im SPARQL 1.1 Graph Store HTTP Protocol keine Möglichkeit gibt, die vorhandenen Named Graphs zu ermitteln. `RDFStorageImpl` nutzt daher für die Methode `getAllStoredGraphs` die SPARQL 1.1 Query Language, um die URIs der gespeicherten Named Graphs abzufragen. Alle Operationen, die den Triple Store manipulieren und entsprechende Rechte benötigen, werden über das SPARQL 1.1 Graph Store HTTP Protocol ausgeführt. Zur Realisierung der Methode `deleteAll()` greift `RDFStorageImpl` auf die Methode `getAllStoredGraphs()` zurück und löscht die ermittelten Named Graphs einzeln, da mittels des SPARQL 1.1 Graph Store HTTP Protocols nur einzelne Named Graphs gelöscht werden können. Für jede in RDF umgewandelte Ressource des Repositoriums (Item, Collection oder Community) wird ein Named Graph im Triple Store angelegt. Als URI des Named Graphs wird der URI verwendet, der in den RDF Daten für die jeweilige Ressource genutzt wird.

Bei der Implementation trat ein Problem mit der Programmbibliothek Jena auf. Beim Abruf der Daten aus dem Triple Store nutzt Jena bis einschließlich zu Version 2.11.1 einen HTTP-Accept-Header, der N-Triple höher priorisiert als andere RDF-Serialisierung. In N-Triples werden URIs jedoch immer in voller Länge angegeben, während andere Serialisierungen Mechanismen zur Verwendung gekürzter URIs enthalten. Auch wenn der Triple Store Informationen über die Nutzung gekürzter URIs enthielt, gingen diese bei der Umwandlung in N-Triples verloren und wurden

entsprechend nicht geladen. Auf einen Bug-Report vom Autor dieser Arbeit reagierten die Entwickler von Jena schnell: Ab Version 2.11.2 wird Jena im HTTP-Accept-Header Turtle höher priorisieren als N-Triples.[167] Als Workaround bis zum Erscheinen von Jena 2.11.2 wurde für `dspace-rdf` die Klasse `org.dspace.rdf.storage.CustomDatasetGraphAccessorHttp` entwickelt, die von der entsprechenden Klasse in Jena erbt, den HTTP-Accept-Header überschreibt und in `dspace-rdf` an Stelle der entsprechenden Jena Klasse verwandt wird. Nach dem Erscheinen von Jena 2.11.2 sollte der Workaround entfernt werden, um den Umstieg auf folgende Jena-Versionen zu vereinfachen.

4.3.4 Die Generierung von URIs

Das Interface `URIGenerator` wird in Abbildung 4.4 dargestellt. Durch die Bereitstellung verschiedener Klassen zur Umsetzung des Interfaces können beliebige URIs erzeugt und so die Art und Form der generierten URIs bestimmt werden. Damit ist nicht die Schaffung neuer URIs gemeint, sondern das Zusammensetzen von Zeichenketten, die den URIs entsprechen, die das Repositorium zur Referenzierung der Ressourcen nutzt. Beispielsweise kann es nötig sein, Persistent Identifiers in eine bestimmte Form umzuwandeln oder einen URI aus statischen und dynamischen Bestandteilen zusammenzusetzen, wie zum Beispiel der Domain des Repositoriums und einem fixen Pfad als festen und dem Typ und der ID einer Ressource als dynamischen Bestandteilen eines URIs. Unter Umständen können Klassen zur Umsetzung des Interfaces einfach vorhandene Methoden des Repositoriums zur Generierung von URIs aufrufen.

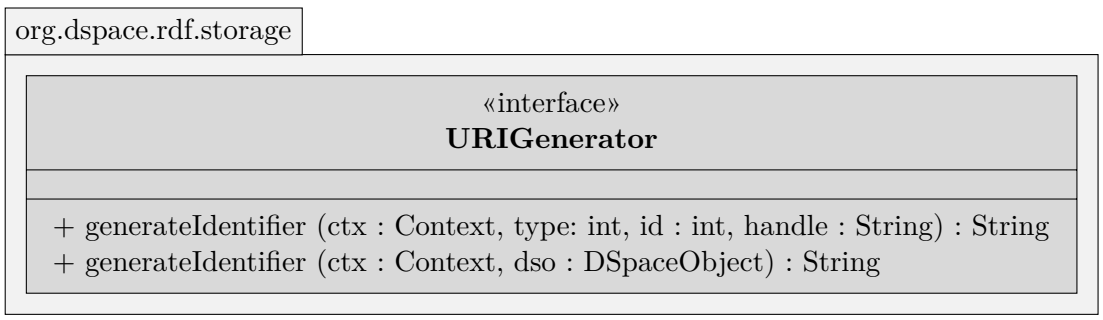

Abbildung 4.4 – Klassendiagramm des Interfaces URIGenerator.

Das Interface schreibt zwei Methoden vor, die beide einen URI generieren und sich lediglich in den Parametern unterscheiden. Beim Löschen von Items, Collections und Communities in DSpace stehen dem `RDFConsumer` nicht alle Informationen

[167] Vgl. https://issues.apache.org/jira/browse/JENA-663, abgerufen am 06.06.2014.

zum gelöschten `DSpaceObject` zur Verfügung. Wie auf S. 64 geschildert, werden Events erst ausgelöst, wenn die Änderungen in die Datenbank geschrieben wurden. Wenn ein Event aufgrund der Löschung einer Ressource ausgelöst wird, kann die Ressource also bereits nicht mehr geladen werden. Das Event enthält jedoch den Typ (Item, Collection, Community, ...), die interne ID und gegebenenfalls das Handle. Um die Ressource auch im Triple Store zu löschen, muss die URI ermittelt werden, die die Ressource in RDF identifiziert. Daher schreibt das Interface `URIGenerator` eine Methode vor, die den URI anhand der im Event enthaltenen Informationen generieren soll. Die Methode, die einen URI anhand der entsprechenden Instanz des `DSpaceObject`s generiert, erleichtert lediglich den Aufruf der URI-Generierung. In dieser Implementation lädt sie den Typ, die interne ID und das Handle aus dem `DSpaceObject` und ruft mit diesen Parametern die andere Methode auf.

Im Rahmen der Implementation wurden dafür zwei Klassen entwickelt: `LocalURIGenerator` erzeugt URIs unterhalb der Domain, die für das Repositorium genutzt wird. `HandleURIGenerator` wandelt Handles in funktionale HTTP-URIs um, das heißt in die Form http://hdl.handle.net/<handle>. Es ist konfigurierbar, welche Klasse für die Generierung der URIs verwandt wird.

4.3.5 RDFConverter und Plugins

Wie zur Anbindung des Triple Stores wurde die Konvertierung der Repositorieninhalte nicht nur in einer Klasse umgesetzt, sondern mit Hilfe eines Interfaces, damit die Konvertierungsstrategie einfach ausgetauscht werden kann. Das entsprechende Interface `RDFConverter` ist sehr einfach: Es enthält eine Methode, die ein Objekt der Klasse `Context`[168] und eines der Klasse `DSpaceObject` erhält und eines vom Typ `Model` zurückgibt. Das `Model` enthält die in RDF konvertierten Daten, die dem übergebenen `DSpaceObject` entsprechen. Die Klasse `RDFConverterImpl` implementiert das Interface `RDFConverter`.

Der Konstruktor der Klasse `RDFConverterImpl` liest aus der Konfiguration aus, welche Plugins genutzt werden sollen, und initialisiert die entsprechenden Klassen. Für die Konvertierung prüft die `RDFConverterImpl` jedes initialisierte Plugin und übergibt das `DSpaceObject` den Plugins, die Objekte dieses Typs konvertieren können – als Typ gelten hier Item, Collection, Community oder ein Typ für die Beschreibung des Repositoriums selbst. Die von den Plugins erstellten Daten führt die Klasse `RDFConverterImpl` in einem `Model` zusammen. Dabei ist es wichtig,

[168] Vgl. Abschnitt 4.3.1 ab S. 55.

auch Prefixes[169] aus den konvertierten Daten zu übernehmen, damit diese in RDF-Serialisierungen, die Prefixes unterstützen, genutzt werden können.

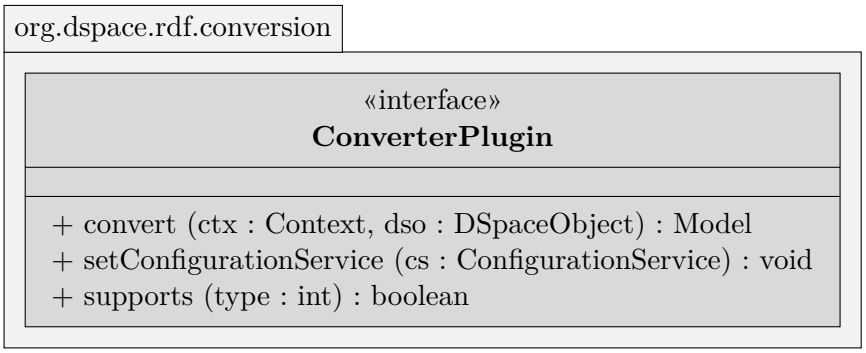

Abbildung 4.5 – Klassendiagramm des Interfaces `ConverterPlugin`. Beim Typ `ConfigurationService` handelt es sich um eine Interface von DSpace, das Zugriff auf die Konfiguration bietet.

Das Interface `ConverterPlugin`, das die API der Plugins für die Konvertierung definiert, ist in Abbildung 4.5 abgebildet. Es enthält drei Methoden. Die Methode `convert(Context, DSpaceObject)` wird aufgerufen, um das beim Aufruf übergebene `DSpaceObject` zu konvertieren. Die Methode `setConfigurationService(ConfigurationService)` dient dazu, dem Plugin eine Referenz auf den DSpace-Service zu geben, über den die Konfiguration ausgelesen werden kann.[170] Und über die Methode `supports(int type)` kann abgefragt werden, ob ein Plugin die Konvertierung eines `DSpaceObject` eines bestimmten Typs unterstützt.

Insgesamt wurden drei Plugins zur Konvertierung der Daten in RDF erstellt. Das `StaticDSOConverterPlugin` lädt für jeden Typ eines `DSpaceObject`s eine RDF-Konfigurationsdatei, deren Inhalt es direkt in die Beschreibung des `DSpaceObject`s übernimmt. Damit lassen sich statische Tripel vorgeben, die in jedem Item, jeder Collection, jeder Community oder der Beschreibung des Repositoriums selbst enthalten sein sollen. Eine weitere Konfigurationsdatei des `StaticDSOConverterPlugin`s ermöglicht es, Tripel vorzugeben, die in jedem Datensatz enthalten sein sollen, unabhängig vom Typ des konvertierten `DSpaceObject`s. Dieses Plugin kann zum Beispiel genutzt werden, um auf den SPARQL-Endpoint hinzuweisen, in dem

[169] Hier und im Folgenden wird bewusst das englische Wort *Prefix* beziehungsweise seine Mehrzahl genutzt, da es zum Beispiel in Turtle als Schlüsselwort für den Mechanismus dient, der es ermöglicht, URIs verkürzt darzustellen.

[170] Die Klasse `RDFConfiguration` vereinfacht das Auslesen der Konfiguration für die Klassen in `dspace-rdf`. Da Plugins zur Erweiterung der Datenkonvertierung ergänzt werden sollen und nicht vorausgesehen werden kann, welche Informationen solche zukünftigen Plugins aus der Konfiguration auslesen wollen, wurde im Interface `ConverterPlugin` auf die Nutzung der Klasse `RDFConfiguration` verzichtet.

die Daten bereitgestellt werden, auf die Lizenz, unter der die Daten stehen, oder auf den Betreiber des Repositoriums. Das `DSORelationsConverterPlugin` bildet die Beziehungen zwischen Items, Collections und Communities in RDF ab, wie es oben im Konzept in Abschnitt 4.2.4 ab S. 55 beschrieben worden ist. Das Plugin selbst ist derzeit als reine Machbarkeitsstudie umgesetzt und daher das einzige Plugin, in dem die verwendeten RDF-Vokabulare noch nicht konfigurierbar sind. Das dritte entwickelte Plugin, das `MetadataConverterPlugin`, wandelt die Metadaten der Items in RDF um. Aufgrund des Umfangs des Plugins und seiner zentralen Bedeutung für die Implementation wird es im folgenden Abschnitt ausführlich beschrieben.

4.3.6 Plugin zur Konvertierung der Metadaten

Das `MetadataConverterPlugin` dient der Konvertierung der Metadaten von Items in RDF. DSpace unterstützt derzeit Metadaten nur für Items, es gibt jedoch Pläne DSpace dahingehend zu erweitern, dass Metadaten für Collections und Communities unterstützt werden. Das Plugin dürfte sich ohne großen Aufwand auch auf Collections und Communities anwenden lassen. Allerdings ist das abhängig von der genauen Umsetzung der Erweiterung von DSpace, speziell von den Methoden für den Zugriff auf die Metadaten der Communities und Collections.

Das `MetadataConverterPlugin` nutzt zur Konvertierung eine RDF-Konfigurationsdatei, die aus beliebig vielen sogenannten Mappings besteht. Jedes Mapping beschreibt ein Metadatenfeld und eine beliebige Anzahl von Tripeln, die erstellt werden sollen, wenn das Metadatenfeld in RDF umgewandelt wird. Dazu wurde ein einfaches Vokabular entwickelt.[171] Ein Beispiel für eine solche Konfiguration ist in Abbildung 4.6 auf der nächsten Seite zu sehen.

Ein Metadatenfeld wird anhand seines Namens ausgewählt. Zusätzlich kann ein regulärer Ausdruck spezifiziert werden. Die für das Metadatenfeld angegebenen Tripel werden dann nur erzeugt, wenn der Wert des Metadatenfeldes den regulären Ausdruck erfüllt. Prinzipiell werden die zu erzeugenden Tripel mittels Reification[172] spezifiziert. Das Vokabular enthält eine Ressource, die als Platzhalter für den URI dient, der das aktuell zu konvertierende Item in RDF identifiziert. Darüber hinaus kann der Wert des Metadatenfeldes durch einen regulären Ausdruck angepasst und zur Generierung von URIs oder Literalen verwandt werden. Zu einigen Metadatenfeldern speichert DSpace eine Sprachangabe, so dass zum Beispiel ein Titel in

[171] Das Vokabular ist Teil des Quellcodes der Implementation und steht unter http://www.pnjb.de/rdf/namespace/dspace/metadata-rdf-mapping/0.2.0 zum Abruf bereit. Mittels Content Negotiation ist das Vokabular in den Serialisierungen RDF/XML, Turtle und N-Triples abrufbar.

[172] Vgl. Fußnote 138 auf S. 56.

```
@prefix dc:      <http://purl.org/dc/elements/1.1/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix bibo:    <http://purl.org/ontology/bibo/> .
@prefix dm:
    <http://www.pnjb.de/rdf/namespace/dspace/metadata-rdf-mapping/0.2.0#>
    .

@prefix :        <#> .

:title
  dm:metadataName "dc.title" ;
  dm:creates [
    dm:subject dm:DSpaceObjectIRI ;
    dm:predicate dcterms:title ;
    dm:object dm:DSpaceValue ;
  ] ;
  .

:languageISO
  dm:metadataName "dc.language.iso" ;
  dm:creates [
    dm:subject dm:DSpaceObjectIRI ;
    dm:predicate dc:language ;
    dm:object [
      a dm:LiteralGenerator ;
      dm:modifier [
        dm:matcher "\^(..)\_(.*)$" ;
        dm:replacement "$1-$2";
      ];
      dm:pattern "$DSpaceValue";
    ];
  ];
  .

:localURIToHandle
  dm:metadataName "dc.identifier.uri" ;
  dm:condition "^http://localhost:8080/jspui/handle/" ;
  dm:creates [
    dm:subject dm:DSpaceObjectIRI ;
    dm:predicate bibo:handle ;
    dm:object [
      a dm:ResourceGenerator ;
      dm:modifier [
        dm:matcher "^http://localhost:8080/jspui/handle/(.*)$" ;
        dm:replacement "http://hdl.handle.net/$1";
      ];
      dm:pattern "$DSpaceValue";
    ];
  ];
  .
```

Abbildung 4.6 – Eine beispielhafte Konfiguration des Metadata-Converter-Plugins.

mehreren Sprachen angegeben werden kann. Eine solche Sprachangabe kann in RDF als Sprachangabe eines Literals übernommen werden. Bei der Implementation wurde darauf geachtet, Sprach- und Typangaben der Reified Statements zu übernehmen, so dass sie auch statisch angegeben werden können.

Abbildung 4.6 auf der vorherigen Seite stellt beispielhaft eine Konfiguration des `MetaDataConverterPlugin`s dar, die die Umwandlung der drei Metadatenfelder `dc.title`, `dc.language.iso` und `dc.identifier.uri` beschreibt. Zu jedem Metadatenfeld `dc.title` soll demnach ein Tripel erstellt werden, das der URI des umzuwandelnden `DSpaceObject`s als Subjekt enthält, da der im Vokabular vorgesehene Platzhalter für das Subjekt verwendet wird. Als Prädikat wird der URI http://purl.org/dc/terms/title statisch gesetzt. Als Objekt wird ein weiterer Platzhalter des Vokabulars verwandt, der durch ein Literal ersetzt wird, das aus dem unveränderten Wert des Metadatenfeldes besteht.

Das Metadatenfeld `dc.language.iso` wird ähnlich umgesetzt, allerdings wird ein anderer URI als Prädikat verwendet und der Wert des Metadatenfeldes gegebenenfalls verändert, ehe ein Literal generiert wird. Dazu sind drei Angaben notwendig: ein regulärer Ausdruck, den die Bereiche des Metadatenwertes erfüllen müssen, die ersetzt werden sollen, ein regulärer Ausdruck, der diese Bereiche ersetzt, und eine Angabe, wo die so erzeugte Zeichenkette eingesetzt werden soll. Beginnt der Wert des Metadatenfedes in unserem Beispiel aus zwei beliebigen Zeichen, gefolgt von einem Unterstrich, so wird der Unterstrich in einen Bindestrich umgewandelt. Dadurch wird aus einer Sprachangabe wie „en_GB“ die Sprachangabe „en-GB“. Erfüllt der Wert des Metadatenfeldes den ersten regulären Ausdruck nicht, wird der Wert unverändert übernommen.

Im dritten Beispiel wird ein weiterer regulärer Ausdruck angegeben. Die spezifizierten Tripel werden nur erzeugt, wenn dieser zusätzliche reguläre Ausdruck durch den Wert des Metadatenfeldes erfüllt wird. Das Metadatenfeld wird also nicht nur abhängig vom Namen, sondern auch vom Wert ausgewählt. Im Beispiel ist der reguläre Ausdruck erfüllt, wenn der Wert mit „http://localhost:8080/jspui/handle/“ beginnt. Das erzeugte Tripel enthält als Subjekt den URI des `DSpaceObject`s, als Prädikat den URI http://purl.org/ontology/bibo/handle und als Objekt einen URI, der mit http://hdl.handle.net/ beginnt, gefolgt von einem Teil des Wertes des ursprünglichen Metadatenfeldes. Konkret erzeugt dieses dritte Beispiel ein Tripel, das angibt, dass das `DSpaceObject` über das im Objekt angegebene Handle adressiert werden kann, wobei es das Handle aus der in DSpace genutzten Adresse ableitet.

Zur Umsetzung der regulären Ausdrücke werden in dieser Implementation die Klassen `Pattern` und `Matcher` aus der Java-API genutzt. Die Notation der

regulären Ausdrücke muss daher der dort genutzten Notation entsprechen. Dafür steht der volle Umfang beider Klassen zur Verfügung, inklusive der in diesen Klassen berücksichtigten *Embedded Flags*[173].

Das `MetadataConverterPlugin` wird unter Nutzung dreier Hilfsklassen realisiert, die im Klassendiagramm in Abbildung 4.7 auf der nächsten Seite dargestellt sind. Das Vokabular DSpace Metadata RDF Mapping[174], das für die Konfiguration des Plugins entworfen wurde, wird von der Klasse `DMRM` in Objekte der Programmbibliothek Jena umgesetzt, was die Handhabung im restlichen Code vereinfacht. Diese Klasse ist im Klassendiagramm gekürzt dargestellt, da sie viele Attribute enthält, die auch aus dem Vokabular selbst abgelesen werden können.

Das `MetadataConverterPlugin` lädt bei der Initialisierung die Konfiguration. Die dazu genutzten Schlüssel sind als Klassenattribute implementiert, was eine Änderung der Konfigurationsschlüssel vereinfacht. Nachdem die Konfiguration geladen wurde, nutzt das Plugin einfache Ableitungsregeln, um bestimmte Entitäten in der Konfiguration zu erkennen. Dadurch müssen etliche Tripel mit dem Prädikat `rdf:type` nicht in die Konfiguration aufgenommen werden, sondern können aus der Nutzung bestimmter Prädikate in der Konfiguration abgeleitet werden. Typangaben sind so nur nötig, um sicher zwischen Generatoren für Literale und URIs unterscheiden zu können. Neben der Konfiguration lädt das `MetadataConverterPlugin` eine Datei, die die Prefixes enthält, die später genutzt werden sollen. Dadurch können die Prefixes zentral festgelegt werden.

Für jedes Mapping in der Konfiguration wird ein Objekt der Klasse `MetadataRDFMapping` initialisiert. Für jedes Metadatenfeld des Items, das konvertiert werden soll, werden alle initialisierten Mappings dahingehend überprüft, ob der angegebene Feldnamen mit dem des Metadatenfeldes übereinstimmen und ob gegebenenfalls die regulären Ausdrücke vom Wert des Feldes erfüllt werden. Die spezifizierten Tripel aller passender Mappings werden dann von den Objekten der Klasse `MetadataRDFMapping` erzeugt.

Für die Felder, für die keine passendes Mapping gefunden wurde, werden alle Mappings ein zweites Mal überprüft. Bei dieser zweiten Prüfung werden Qualifier in den Namen der Felder ignoriert. Grundlage dafür ist, dass DSpace in der Standardkonfiguration ein Metadatenschema nutzt, das an Qualified Dublin Core angelehnt ist. In Qualified Dublin Core können zusätzlich zu Feldnamen auch Qualifier angegeben, die den Kontext eines Feldes näher umreißen sollen. Prinzipiell sollten Mappings, für deren Feld kein Qualifier angegeben wurde, auch

[173] Über sogenannte `Flags` lässt sich die Interpretation eines regulären Ausdrucks verändern. In der Dokumentation der beiden Klassen wird von `Embedded Flags` gesprochen, wenn Flags innerhalb eines regulären Ausdrucks und nicht über den Aufruf von Methoden kontrolliert werden.

[174] Vgl. Fußnote 171 auf Seite 69.

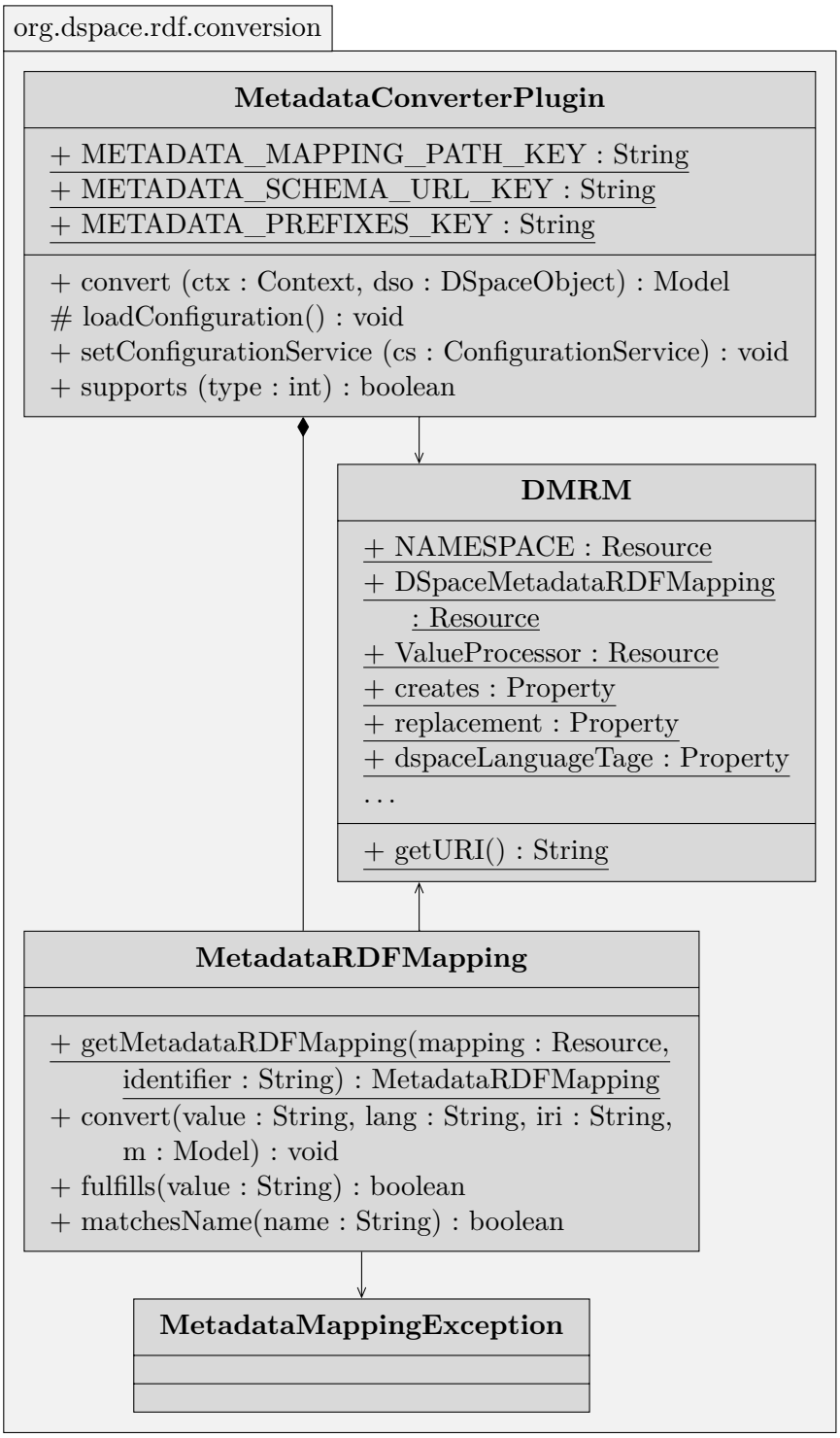

Abbildung 4.7 – Vereinfachtes Klassendiagramm des MetadataConverterPlugins und der zugehörigen Hilfsklassen.

auf Felder anwendbar sein, deren Name jeweils mit einem Qualifier angegeben ist. Das heißt zum Beispiel, dass ein Feld `dc.language.iso` auch mit einem Mapping für ein Feld `dc.language` konvertierbar sein sollte.[175] Bei der zweiten Prüfung der Mappings ignoriert die Klasse `MetadataConverterPlugin` deshalb die Qualifier der Metadatenfelder, die zuvor nicht konvertiert wurden, und versucht diese Felder erneut zu konvertieren. Mappings, deren Feldnamen Qualifier enthalten, werden dabei ignoriert, damit nicht versehentlich zum Beispiel ein Feld `dc.language` oder ein Feld `dc.language.rfc3306` mit einem Mapping für `dc.language.iso` konvertiert wird.

Die Klasse `MetadataRDFMapping` wirft und fängt intern eine `MetadataMappingException`, falls ein Mapping nicht korrekt erkannt werden kann. Wenn solche oder andere Fehler auftreten, zum Beispiel weil ein regulärer Ausdruck nicht korrekt angegeben wurde, werden entsprechende Hinweise in eine Logdatei geschrieben. Das jeweilige Mapping wird im Fehlerfall ignoriert und die übrige Konvertierung wird fortgesetzt. Das `MetadataConverterPlugin` prüft, ob ein Feld ausgegeben werden darf, bevor es die Konvertierung des Feldes startet. Felder die in der Konfiguration als „versteckt“ angegeben wurden, werden dem Konzept entsprechend nicht konvertiert.

4.3.7 RDF Serialisierungen und Content Negotiation

Die Speicherung der konvertierten Daten im Triple Store wirkt sich vor allem für die Bereitstellung der Daten als RDF-Serialisierung positiv aus. Da die Daten bereits konvertiert sind, müssen sie lediglich aus dem Triple Store abgerufen und in das angeforderte Format gebracht werden. Darüber hinaus können die im Triple Store gespeicherten Daten bereitgestellt werden, ohne zu prüfen, ob sie öffentlich zugänglich sein sollen, da eine solche Prüfung bereits während der Konvertierung stattfand. Im Rahmen der Implementation wurde das `DataProviderServlet` entwickelt, das diese Aufgaben übernimmt.

Die Content Negotiation ist dem `DataProviderServlet` vorgelagert. In den in RDF konvertierten Daten werden URIs verwandt, die bereits im Repositorium genutzt und auf die Weboberfläche des Repositoriums weitergeleitet werden. Bei der Dereferenzierung dieser URIs generieren je nach Konfiguration `dspace-jspui` oder `dspace-xmlui` HTML, das zurück geliefert wird. Damit die URIs gemäß der Linked Data Principles verwendet werden können, müssen `dspace-jspui` und `dspace-xmlui` um Content Negotiation erweitert werden. Dadurch wird vor der

[175] Dublin Core bezeichnet diese Herangehensweise als „Dump-Down-Principle“. Vgl. http://dublincore.org/resources/faq/#dumbdown, abgerufen am 06.06.2014.

Auslieferung von HTML anhand des HTTP-Accept-Headers entschieden, ob HTML oder eine Serialisierung der in RDF konvertierten Daten zurückgesendet werden soll. Im Rahmen der Implementation wurde der `NegotiationFilter` entwickelt, der als Servlet Filter in `dspace-jspui` eingebunden werden kann. Wird zum Beispiel die Jump-Off-Page eines Items abgefragt, führt der `NegotiationFilter` die Content Negotiation durch. Wird dabei im HTTP-Accept-Header RDF/XML, N-Triples oder Turtle mit einer höheren Priorität als alle anderen angegebene Formate angefordert, leitet der `NegotiationFilter` auf den entsprechenden URI des `DataProviderServlet`s um. Wird nicht nach einer der genannten RDF-Serialisierungen gefragt, lässt der `NegotiationFilter` die Anfrage unverändert passieren, und `dspace-jspui` arbeitet die Anfrage ab wie bereits vor der Ergänzung des Filters.

Der `NegotiationFilter` wurde für `dspace-jspui` entwickelt. In `dspace-xmlui` kommen die Frameworks Spring und Cocoon zum Einsatz, um zu entscheiden, welche Komponenten zur Bearbeitung einer Anfrage genutzt werden. Wahrscheinlich lässt sich der `NegotiationFilter` auch in `dspace-xmlui` nutzen. Allerdings wäre es besser, die Content Negotiation für `dspace-xmlui` unter Nutzung der genannten Frameworks umzusetzen. Bei Bedarf kann dabei die Klasse `MediaRange` wiederverwendet werden, die den HTTP-Accept-Header analysiert. Im Rahmen der Proof of Concept Implementation wurde auf eine Umsetzung für `dspace-xmlui` verzichtet, da keine konzeptionellen Unterschiede zu erwarten sind und mittels des `NegotiationFilter`s in der Proof of Concept Implementation gezeigt wurde, dass sich Content Negotiation in DSpace integrieren lässt.

Werden Persistent Identifiers in Form funktionaler HTTP-URIs verwendet, wird beim Dereferenzieren der funktionalen URIs ein Resolver-Dienst für jeden Persistent Identifier aufgerufen. Dieser leitet auf den URI weiter, unter dem die Inhalte zu finden sind, die durch den Persistent Identifier adressiert werden. Im Fall von Repositorien ist dies ein URI der Weboberfläche, bei dessen Dereferenzierung der `NegotiationFilter` einspringt und gegebenenfalls zum `DataProviderServlet` weiterleitet. Werden Serialisierungen der in RDF konvertierten Daten über Persistent Identifier abgerufen, gibt es also zwei Weiterleitungen. Einige Resolver für Persistent Identifiers beherrschen Content Negotiation. Sie können bei entsprechenden Anfragen auch direkt auf das `DataProviderServlet` weiterleiten, wenn sie entsprechend konfiguriert wurden. In Bezug auf DSpace trifft das zum Beispiel auf den Resolver für DOIs zu. Allerdings müsste noch der Code in DSpace angepasst werden, der DOIs registriert, so dass er die Content Negotiation des Resolvers aktiviert und Informationen zu den verschiedenen URIs an den Resolver gesendet werden. Ehe DOIs von DSpace zur Identifikation von Ressourcen in RDF genutzt werden, sollte das auf Seite 64 diskutierte Problem gelöst werden.

In DSpace werden Adressen von Items, Collections und Communities unter Nutzung des Handles des jeweiligen Objekts vergeben. Dieser Adresskonvention folgt das Servlet. An diesen Teil der Adresse wird noch eine Dateiendung der gewünschten Serialisierung angehängt. Eine solche Adresse sieht zum Beispiel wie folgt aus: http://dspace.pnjb.de/rdf/handle/123456789/1/ttl. Anhand des Handles erkennt das `DataProviderServlet`, von welchem Item, welcher Collection oder Community die Daten angefragt wurden. Über die Klasse `RDFUtil` und den konfigurierten `URIGenerator` generiert es den URI, der für das jeweilige Objekt in RDF verwandt wird. Anschließend lädt es über das `RDFUtil` und den konfigurierten `RDFStorage` die konvertierten Daten und liefert sie in der gewünschten Serialisierung aus.

4.3.8 Installation, Konfiguration und Nutzung

Jede DSpace Version erscheint als so genanntes *Source Release* und als *Binary Release*. Der Quellcode von DSpace enthält für jede Hauptversion einen Zweig. `dspace-rdf` setzt auf den Zweig der aktuell in Entwicklung befindlichen Version 5 von DSpace auf, dennoch kann die Installationsanleitung für das Source Release der Version 4[176] genutzt werden. Anstatt das offizielle Source Release von DSpace zu laden, muss das tar-Archiv[177] genutzt werden, das den DSpace-Quellcode samt `dspace-rdf` enthält.

Nach der Installation von DSpace muss ein Triple Store bereitgestellt werden. Zur Unterstützung von Repositorienbetreibern enthält `dspace-rdf` eine Konfigurationsdatei für Fuseki[178], die unter `[dspace-install]/config/modules/rdf/fuseki-assembler.ttl` abgelegt wird, wobei `[dspace-install]` durch den Pfad zum Installationsverzeichnis von DSpace ersetzt werden muss. Fuseki wird im Rahmen von Jena entwickelt, nutzt Jena als Triple Store und bietet einen SPARQL-Endpoint.[179] Da die Konfigurationsdatei Fuseki so konfiguriert, dass eine Manipulation des Triple Stores über das SPARQL 1.1 Graph Store HTTP Protocol möglich ist, wird dringend empfohlen Fuseki mit der Option `--localhost` zu starten. Diese Option sorgt dafür, dass Fuseki nur von dem Rechner aus erreicht werden kann, auf dem es läuft, damit der Triple Store nicht von jeder beliebigen Adresse manipuliert werden kann. Mittels Apache und den Modulen `proxy`, `proxy-http` und `rewrite`

[176] Siehe dazu https://wiki.duraspace.org/display/DSDOC4x/Installing+DSpace, abgerufen am 06.06.2014.

[177] Dieses tar-Archiv liegt der Diplomarbeit auf CD bei und kann nach Abschluss des Prüfungsverfahrens unter http://www.pnjb.de/uni/diplomarbeit/ herunter geladen werden.

[178] Vgl. Abschnitt 4.2.1 ab S. 50.

[179] Vgl. http://jena.apache.org/documentation/serving_data/index.html, abgerufen am 06.06.2014.

lässt sich ein allgemeiner Zugriff auf den SPARQL-Endpoint einrichten, der nur für den lesenden Zugriff konfiguriert ist.

Die Implementation selbst kann konfiguriert werden, indem die Datei `[dspace-install]/config/modules/rdf.cfg` und die Dateien im Verzeichnis `[dspace-install]/config/modules/rdf/` angepasst werden. In der Konfigurationsdatei `[dspace-install]/config/dspace.cfg` muss der `RDFConsumer` aktiviert werden. Dazu muss der Wert des Konfigurationsschlüssels `event.dispatcher.default.consumers` – den Kommentaren in der Konfigurationsdatei entsprechend – um die Zeichenkette `rdf` erweitert werden. Die Implementation enthält eine grundlegende Konfiguration, die bereits die wichtigsten Metadatenfelder konvertiert. Wenn Inhalte in DSpace aufgenommen, gelöscht oder verändert werden, muss der Triple Store erreichbar sein, da DSpace nach der Änderung der Datei `dspace.cfg` versucht, Änderungen im Triple Store nachzuvollziehen. Ist der Triple Store nicht erreichbar, werden die Änderungen zwar im Repositorium, jedoch nicht im Triple Store umgesetzt und eine entsprechende Fehlermeldung in die Log-Datei von DSpace geschrieben.

Durch den Befehl `[dspace-install]/bin/dspace dsrun org.dspace.rdf.RDFizer --help` kann das Command-Line-Interface genutzt werden. Unter Windows ist anstelle von `[dspace-install]/bin/dspace` wie üblich `[dspace-install]/bin/dspace.bat` zu nutzen. Die Option `--help` startet die Online-Hilfe, die weitere Informationen über die Nutzung des Command-Line-Interface gibt.

Kapitel 5

Evaluation

Zur Evaluation des vorgestellten Konzepts werden in diesem Kapitel zwei Ansätze verfolgt. Zum einen wird das Konzept anhand der Zielsetzung aus Abschnitt 4.1 evaluiert. Zum anderen wird am Beispiel der Proof of Concept Implementation betrachtet, wie sich das Konzept umsetzen lässt. Da nur wenige Punkte fehlen, um die Implementation als produktreife Entwicklung anzusehen, werden diese in einem dritten Abschnitt kurz benannt.

5.1 Evaluation anhand der Zielsetzung

In Abschnitt 4.1 ab S. 47 wurden sieben Anforderungen an das Konzept formuliert. In diesem Abschnitt soll das Konzept dahingehend überprüft werden, in wie weit es diese Anforderungen erfüllt.

1. Berücksichtigung der Linked Data Principles

   Im Konzept ist vorgesehen, URIs, die das Repositorium bereits nutzt, wiederzuverwenden und Persistent Identifiers in Form funktionaler HTTP-URIs anzugeben.[180] Durch die Verwendung von URIs, die das Repositorium bereits nutzt, ist sichergestellt, dass die URIs dereferenziert werden können und es eine HTTP-Repräsentation gibt. Des Weiteren sieht das Konzept vor, für diese URIs Content Negotiation einzusetzen, um entsprechende Anfragen zu der Komponente weiterzuleiten, die die Informationen als Serialisierung von RDF bereitstellt.[181] Der Triple Store soll die konvertierten Daten zusätzlich

[180] Vgl. Abschnitt 4.2.5 ab S. 57.
[181] Vgl. Abschnitt 4.2.6 ab S. 58.

über einen SPARQL-Endpoint bereitstellen.[182] Damit erfüllt das Konzept die ersten drei Empfehlungen der Linked Data Principles.

Die vierte Empfehlung der Linked Data Principles besteht darin, Daten zu verlinken, wo immer das möglich ist. Im Konzept ist zum einen vorgesehen, Links zwischen den Ressourcen des Repositoriums zu generieren. Dadurch ist es möglich, den RDF-Links folgend das ganze Repositorium zu durchwandern.[183] Zum anderen soll die Konfiguration der Konvertierung der Metadaten die Möglichkeiten bieten, Links aus den Namen und Werten der Metadatenfelder zu generieren.[184] Gerade in Bezug auf die Generierung von Links unterscheidet sich dieses Konzept zum Beispiel von der Bereitstellung von RDF, wie sie in EPrints umgesetzt wurde.[185] Die Linked Data Principles finden im Konzept volle Beachtung.

2. Dopplung von Funktionen vermeiden

   Das Konzept sieht die Verwendung von URIs vor, die das Repositorium bereits eingeführt hat, und nutzt zur HTML-Repräsentation Funktionen des Repositoriums. Auch Persistent Identifiers wurden im Konzept hinreichend berücksichtigt.[186] Die Integration in die API der Repositoriensoftware, die im nächsten Punkt näher diskutiert wird, vereinfacht die Nutzung von Funktionen des Repositoriums in der Umsetzung des Konzepts. Die Dopplung von Funktionen zwischen dem Repositorium und den zu ergänzenden Komponenten wird somit vom Konzept vermieden.

3. Berücksichtigung der softwaretechnischen Kapselung

   Ein direkter Zugriff auf die Datenbank oder das Dateisystem ist im Konzept nicht vorgesehen. Vielmehr fordert das Konzept eine Integration in die API der Repositoriensoftware. Im Konzept ist mit dem `RDFUtil` eine Schnittstelle zwischen den ergänzten Komponenten und dem übrigen Repositorium vorgesehen. Ist in der Repositoriensoftware ein Event-System realisiert, dann sollen Änderungen im Triple Store über eine Einbindung in das Event-System nachvollzogen werden.[187] Zugriffsbeschränkungen, die im Repositorium festgelegt sind, werden vom Konzept beachtet, wozu über die API des Repositoriums überprüft werden muss, ob eine Zugriffsbeschränkung für ein bestimmtes Objekt eingestellt ist, oder nicht.[188] Die softwaretechnische Kapselung der

[182] Vgl. Abschnitt 4.2.1 ab S. 50.
[183] Vgl. Abschnitt 4.2.4 ab S. 55.
[184] Vgl. Abschnitt 4.2.4 ab S. 55.
[185] Vgl. Abschnitt 3.3 ab S. 42.
[186] Vgl. Abschnitte 4.2.5 und 4.2.6 ab S. 57.
[187] Vgl. Abschnitt 4.2.2 ab S. 52.
[188] Vgl. Abschnitte 4.2.1 ab S. 50 und 4.2.3 ab S. 54.

Repositoriensoftware findet sich somit in angemessener Art und Weise im Konzept wieder.

4. Einfache Nutzung durch Repositorienbetreiber

   Bereits in der Zielsetzung wurde darauf hingewiesen, dass dieser Punkt sich eher an die Umsetzung des Konzepts als direkt an das Konzept richtet. Das Konzept berücksichtigt, dass die Konvertierung umfassend konfigurierbar sein sollte, was die Anpassung durch Repositorienbetreiber erleichtert.[189] Dadurch dass die Daten in RDF ergänzend gespeichert werden und ein eventuell verwendetes relationales Datenbanksystem erhalten bleibt und nicht durch einen Triple Store ersetzt wird, schafft das Konzept die Voraussetzung für eine leichte Migration auf eine Version der Repositoriensoftware, in der dieses Konzept umgesetzt wurde.

   Im Zusammenhang mit der Zielsetzung einer einfachen Nutzung durch Repositorienbetreiber sei auch darauf verwiesen, dass sich das Berufsbild des Bibliothekars – Repositorien werden häufig von Bibliotheken betrieben – aktuell wandelt und zunehmend technische Aufgaben integriert.[190] Damit einhergeht der zunehmende Aufbau technischer Kompetenz in Bibliotheken.

5. Flexibilität und umfassende Konfigurierbarkeit

   In Bezug auf die Flexibilität und die Konfigurierbarkeit gibt es zwei Ansätze im Konzept. Zum einen ist vorgesehen, Plugins zur Konvertierung der Repositorieninhalte zu nutzen, so dass weitere Plugins einfach ergänzt werden können.[191] Zum anderen sollen die Plugins konfigurierbar sein. Mittels regulärer Ausdrücke, die während der Konvertierung auf den Wert eines Metadatenfeldes angewandt werden, sollen URIs und Literale erzeugt werden. Damit lässt sich einfach definieren, wie welche Metadatenfelder umgesetzt, welche Vokabulare dabei verwandt und wie Links generiert werden sollen.[192] Durch eine solche Konfiguration wird im Konzept sichergestellt, dass die zu konvertierenden Metadatenfelder und die verwendeten Vokabulare angepasst werden können, ohne dass der Quellcode verändert werden muss. Dadurch dass in der Konfiguration ausgewählt werden kann, welche Plugins zur Konvertierung und welcher Algorithmus zur URI-Generierung verwandt werden, erhalten Repositorienbetreiber zusätzliche Möglichkeiten, die Konvertierung einfach an ihr Repositorium anzupassen. Auch anhand der Implementation wird ersichtlich, wie mächtig die Mechanismen des Konzepts in Bezug auf Flexibilität und Konfiguration sind.

[189] Vgl. Abschnitt 4.2.4 ab S. 55.
[190] Vgl. bezüglich des sich wandelnden Berufsbilds zum Beispiel Becker und Fürste 2013.
[191] Vgl. Abschnitt 4.2.3 ab S. 54.
[192] Vgl. Abschnitt 4.2.4 ab S. 55.

6. Konvertierung oder Verlinkung der digitalen Objekte

   In der Zielsetzung wird dargelegt, wieso es wichtig ist, die in den Repositorien gespeicherten digitalen Objekte sowohl konvertieren als auch verlinken zu können. Durch die Nutzung von Plugins, die die eigentliche Konvertierung vornehmen, kann das erreicht werden.[193] Denkbar ist zum Beispiel ein Plugin, das CSV-Dateien in RDF konvertiert, oder ein Plugin, das Bilder verlinkt. Darüber hinaus wird im Konzept berücksichtigt, dass eine Entscheidung, ob Dateien verlinkt oder konvertiert werden, oft nur sinnvoll zu treffen ist, wenn alle Metadaten und alle Dateien eines Items zur Konvertierung vorliegen.[194] Es reicht in der Regel nicht aus, die Dateien voneinander losgelöst zur Konvertierung vorzulegen. Im Konzept ist also vorgesehen, dass Plugins entwickelt werden, die digitale Objekte sowohl verlinken als auch konvertieren können.

7. Erweiterbarkeit des Konzepts und seiner Umsetzung

   Das Konzept sieht vor, dass mehrere Plugins zur Konvertierung der Daten in RDF und diverse Algorithmen zur Generierung von URIs umgesetzt werden. Es soll konfigurierbar sein, welche Plugins zum Einsatz kommen und welcher Algorithmus zur Generierung von URIs genutzt wird.[195] Da mehrere Plugins und Algorithmen zur Generierung von URIs im Konzept bereits vorgesehen sind, ist es einfach, weitere hinzuzufügen. Durch neue Plugins und andere Algorithmen zur Generierung von URIs kann das Konzept einfach an Lösungen für spezielle Repositorien angepasst werden.

Die in der Zielsetzung formulierten Anforderungen werden somit alle vom Konzept erfüllt. Die einfache Nutzung für Repositorienbetreiber ist im Konzept berücksichtigt, jedoch mehr von der Umsetzung des Konzepts, als vom Konzept selber abhängig.

5.2 Evaluation anhand der Implementation

Die Proof of Concept Implementation, die in Abschnitt 4.3 dokumentiert wurde, ist eine direkte Umsetzung des Konzepts. Sie fügt die im Konzept geforderten Elemente zu DSpace hinzu. Konkret umgesetzt wurden:

- die Anbindung eines Triple Stores über SPARQL 1.1,

[193] Vgl. Abschnitt 4.2.4 ab S. 55.
[194] Vgl. Abschnitt 4.2.4 ab S. 55.
[195] Vgl. Abschnitte 4.2.3 ab S. 54 und 4.2.5 ab S. 57.

- eine Schnittstelle in Richtung des restlichen Repositoriums,
- die Einbindung in das Event-System von DSpace,
- Mechanismen, um per Konfiguration festlegen zu können, welche Klassen zur Umsetzung einzelner Komponenten geladen werden,
- ein Command-Line-Interface für Administratoren,
- zwei Algorithmen zur Erzeugung von URIs,
- die Steuerung der Konvertierung,
- drei Plugins für die Konvertierung,
- ein Vokabular zur Konfiguration der Konvertierung der Metadaten in RDF,
- ein Servlet zur Auslieferung der Daten als RDF-Serialisierung sowie
- ein Servlet Filter für die Content Negotiation.

Herzstück der Implementation – im Sinne des Proof of Concept – ist die Komponente zur Konvertierung der Metadaten in RDF, insbesondere das `MetadataConverterPlugin` und das Vokabular zu seiner Konfiguration. Mit diesem Plugin wurde gezeigt, dass die im Konzept beschriebene Konfiguration in RDF unter Nutzung von regulären Ausdrücken flexibel ist und gut dazu genutzt werden kann, um aus Namen und Werten einzelner Metadatenfelder Literale und URIs zu generieren.[196] Die in Abbildung 4.6 auf Seite 70 dargestellte beispielhafte Konfigurationsdatei des Plugins gibt einen Eindruck von der Einfachheit der Angabe von Regeln zur Konvertierung von Metadatenfeldern oder zur Veränderung der in den erzeugten Triplen genutzten Vokabulare. Die Konfiguration, die der Implementation hinzugefügt wurde, ist umfassender als das abgebildete Beispiel. Weitere Regeln zur Konvertierung lassen sich nach Belieben ergänzen.

Die im Konzept angesprochenen Vorteile einer Konfiguration in RDF,[197] konkret die Nutzung von Sprach- und Typangaben für Literale, konnte anhand der Implementation positiv verifiziert werden.[198] RDF zur Konfiguration des `MetadataConverterPlugin`s zu nutzen, bringt einen weiteren Vorteil mit sich, der im Konzept nicht angesprochen wurde: Durch den Einsatz logischer Schlussfolgerungen beim Laden der Konfiguration müssen die meisten Ressourcen nicht explizit typisiert[199] werden, was das Erstellen der Konfiguration vereinfacht.[200]

Hinsichtlich der Konfiguration der zu nutzenden Komponenten wurde in der Implementation zu Gunsten einfach anzupassender Konfigurationsdateien auf den Einsatz des Frameworks Spring verzichtet.[201] Die Hauptkonfigurationsdatei enthält

[196] Vgl. Abschnitt 4.3.6 ab Seite 69.
[197] Vgl. Abschnitt 4.2.4 ab S. 55.
[198] Vgl. Abschnitt 4.3.6 ab S. 69.
[199] Mit Typisierung sind in diesem Fall Tripel in RDF gemeint, die als Prädikat `rdf:type` nutzen.
[200] Vgl. Abschnitt 4.3.6 ab S. 69.
[201] Vgl. Abschnitt 4.3.2 ab S. 62.

eine kommaseparierte Liste aller Klassen, die als Plugins bei der Konfiguration zum Einsatz kommen. Die Auswahl des zu nutzenden Algorithmus zur Generierung der URIs, die zur Identifikation der Ressourcen des Repositoriums in RDF genutzt werden, sowie die Auswahl der Klasse, die die Konfiguration steuert, erfolgen genauso einfach unter Angabe der jeweiligen Klassennamen. Die Hauptkonfigurationsdatei findet sich unter `[dspace-install]/config/modules/rdf.cfg`, die Konfigurationsdateien der Plugins werden im Ordner `[dspace-install]/config/modules/rdf/` gebündelt, so dass es leicht ist, sich einen Überblick über die gesamte Konfiguration zu verschaffen.

In der Implementation konnte gezeigt werden, wie Handles in Form funktionaler HTTP-URIs zur Identifikation der Ressourcen in RDF genutzt werden können. Als Alternative können URIs unter Nutzung der Domain, unter der das Repositorium läuft, generiert werden.[202] Ein noch offenes Problem zur Nutzung von DOIs in RDF wurde auf Seite 64 diskutiert. Ein Lösungsvorschlag des Autors liegt der DSpace Community vor. Der Servlet Filter für die Content Negotiation ließ sich problemlos in `dspace-jspui` integrieren, so dass die genutzten URIs im Sinne von Linked Data dereferenziert werden können.[203]

Das Event-System von DSpace konnte genutzt werden, um die Konvertierung von Ressourcen des Repositoriums oder die Löschung von Daten im Triple Store zu starten.[204] Durch die Nutzung des Event-Systems werden Änderungen an Inhalten des Repositoriums unmittelbar im Triple Store nachvollzogen. Für Administratoren steht zusätzlich ein Command-Line-Interface zur Verfügung, über das alle Repositorieninhalte oder gezielt einzelne Items, Collections oder Communities konvertiert und aus dem Triple Store gelöscht werden können.[205]

Die Nutzung von Named Graphs zur Bündelung aller Tripel, die eine Ressource des Repositoriums beschreiben, hat sich in der Implementation bewährt. Sie erleichtert den Abruf aller Tripel einer Ressource zur Auslieferung als RDF-Serialisierung oder das Löschen dieser Tripel, wenn eine Ressource im Repositorium gelöscht oder eine Zugriffsbeschränkung eingeführt wird. Den Zugriffsschutz zum Zeitpunkt der Konvertierung von Daten zu berücksichtigen, hat die Implementation des Servlets zur Auslieferung von Serialisierungen der Daten in RDF sehr vereinfacht. Darüber hinaus ermöglicht dieses Vorgehen, den Triple Store zu nutzen, um einen SPARQL-Endpoint öffentlich bereitzustellen. Ansonsten hätte ein SPARQL-Endpoint erstellt werden müssen, der den Zugriffsschutz berücksichtigt.

[202] Vgl. Abschnitt 4.3.4 ab S. 66.
[203] Vgl. Abschnitt 4.3.7 ab S. 74.
[204] Vgl. Abschnitt 4.3.2 ab S. 62.
[205] Vgl. Abschnitt 4.3.2 ab S. 62 und 4.2.2 ab S. 52.

Zusammenfassend lässt sich festhalten, dass die Proof of Concept Implementation die Umsetzbarkeit des Konzepts nachgewiesen hat. Bei der Implementierung wurde auf eine einfache Nutzung durch Repositorienbetreiber geachtet, insbesondere in Bezug auf die Konfiguration.

5.3 Entwicklung zur Produktreife

Die im Rahmen dieser Arbeit entwickelte Implementation des Konzepts diente dessen Überprüfung. Das Konzept wurde dabei für DSpace vollständig umgesetzt, so dass es sich anbietet die Proof of Concept Implementation bis zur Produktreife weiterzuentwickeln. Dazu bedarf es nur noch weniger Ergänzungen.

Der entscheidende Punkt zum Erlangen der Produktreife ist eine gute Standardkonfiguration. Die Konfiguration, die im Rahmen der Implementation umgesetzt wurde, enthält bereits Regeln zur Umsetzung vieler Metadatenfelder, die DSpace direkt nach einer gewöhnlichen Installation nutzt. Dennoch sollte noch einmal geprüft werden, welche Vokabulare und Ontologien genutzt werden sollen, ob noch Tripel ergänzt werden müssen und wofür noch ein geeignetes Vokabular beziehungsweise eine Ontologie fehlt. Im Rahmen von EPrints wurde die *EPrints Ontology*[206] erstellt. Diese enthält zum Beispiel Properties[207] zum Verlinken einer OAI-PMH-Schnittstelle und der digitalen Objekte eines Repositoriums. Es ist zu prüfen, ob diese Ontologie in DSpace genutzt werden sollte oder ob von dieser Ontologie ausgehend eine Ontologie zur Beschreibung von Repositorien abstrahiert werden kann, die unabhängig von einer speziellen Softwarelösung ist.

Das `DSORelationsConverterPlugin` muss um eine Konfiguration ergänzt werden, über die sich die in den erzeugten Tripeln verwendeten Vokabulare spezifizieren lassen. In der Proof of Concept Implementation wurde das nicht realisiert, da die Konfigurierbarkeit von Plugins bereits prototypisch an anderen Plugins umgesetzt wurde, und es somit nicht der Klärung konzeptioneller Fragen oder der technischen Machbarkeit diente. Eine Konfiguration für das Plugin würde den einfachen Austausch des genutzten Vokabulars ermöglichen.

[206] Vgl. http://www.eprints.org/ontology/, abgerufen am 06.06.2014.

[207] Als *Property* werden URIs bezeichnet, die als Prädikat von Tripeln in RDF genutzt werden. Bei einer Property handelt es sich also um einen URI, der die Relation zwischen Subjekt und Objekt eines Triples spezifiziert. Auch wenn das Wort *Eigenschaft* als Übersetzung von Property aus dem Englischen inhaltlich adäquat erscheint, wird das englische Wort auch im Deutschen als Terminus genutzt und daher hier nicht übersetzt.

Zur Identifikation von Ressourcen in RDF ist die Nutzung von DOIs in Form von HTTP-URIs wünschenswert.[208] Wie auf S. 64 diskutiert, kann das Löschen von Ressourcen aus dem Repositorium im Triple Store allerdings bislang nicht nachvollzogen werden, wenn DOIs zur Identifikation genutzt werden. Sobald dieses Problem innerhalb von DSpace gelöst wurde – eine Lösung wurde vom Autor vorgeschlagen – ist die Erweiterung der Implementation um die Nutzung von DOIs einfach umzusetzen. Optimal wäre es auch die Registrierung von DOIs in DSpace zu erweitern. Wenn DSpace bei der Registrierung von DOIs die URIs hinterlegt, die für die Repräsentation der Ressourcen in HTML, RDF/XML, Turtle und N-Triples genutzt werden, wird die Content Negotiation bereits durch den DOI-Resolver durchgeführt.

Die Weboberflächen von DSpace, konkret die Module `dspace-jspui` und `dspace-xmlui`, sollten erweitert werden, so dass in den HTML-Repräsentationen der Ressourcen auf die Serialisierung der konvertierten Daten in RDF hingewiesen wird. Unit-Tests für `dspace-rdf` würden die Weiterentwicklung erleichtern. Und letztlich wäre es hilfreich, wenn Jena in der Version 2.11.2 erscheinen würde, so dass der Workaround[209] zum Abruf genutzter Prefixes über das SPARQL 1.1 Graph Store HTTP Protocol aus der Implementation entfernt werden kann.

Für die einfache Nutzung durch Repositorienbetreiber wird neben einer guten Standardkonfiguration auch noch eine ausführliche Dokumentation benötigt, die die Konfiguration erklärt und die Anpassung an das jeweilige Repositorium unterstützt. Die Implementation wird der DSpace Community zur Verfügung gestellt, um mit ihr gemeinsam die hier genannten Punkte umzusetzen.

[208] Vgl. 3.1.2 ab S. 30 und 4.2.5 ab S. 57.
[209] Vgl. Abschnitt 4.3.3 ab S. 64.

Kapitel 6

Ausblick

Die vorliegende Arbeit motiviert, konzipiert, implementiert und evaluiert die Einbindung von Repositorien in das Semantic Web. Die dargestellten Resultate werfen eine Vielzahl weiterführender Forschungsfragen auf. Insbesondere ist zu untersuchen, wie Repositorien und das Semantic Web sich gegenseitig bereichern, wie Repositorien von semantischen Technologien profitieren und welche weiteren Schritte zur Integration hilfreich wären.

Durch die Einbindung von Repositorien in das Semantic Web liegen die Repositorieninhalte erstmals in einer Form vor, in der semantische Technologien einfach genutzt werden können. Daraus ergibt sich die Frage, welche Informationen sich zum Beispiel durch logische und regelbasierte Schlussfolgerungen aus den konvertierten Daten gewinnen lassen. Dies impliziert auch Überlegungen, ob und wie die so gewonnenen Informationen in ein Repositorium zurückfließen können.

Links zwischen Ressourcen des Repositoriums bilden in der vorliegenden Arbeit dessen Struktur ab. Wenn zwei Ressourcen auf den selben externen URI verlinken, lassen sich dadurch eventuell weitere Verbindungen zwischen ihnen erkennen. Von großem Interesse sind auch eingehende Links auf die in RDF umgewandelten Daten. Zum Beispiel könnten sie ebenfalls auf Zusammenhänge zwischen verschiedenen Repositorieninhalten hindeuten oder zusätzliche Informationen bereitstellen, die zur Verbesserung der Metadaten genutzt oder durch eine Verlinkung mit den entsprechenden Repositorieninhalten verknüpft werden könnten. Weiterhin steht die prinzipielle Frage im Raum, ob auch Informationen aus dem Semantic Web in das Repositorium zurückgespielt werden sollten.

Die Frage, inwieweit RDF innerhalb von Repositoriensoftware verwandt werden sollte, ist intensiv zu diskutieren. Technisch sind auch Softwarelösungen für Repositorien denkbar, die die Metadaten digitaler Objekte ausschließlich in RDF ablegen

und RDF als internes Datenmodell nutzen. Um die Umsetzung für bestehende Repositorien einfach zu halten, wurde in der vorliegenden Arbeit darauf bewusst verzichtet. Für die Eindeutigkeit von Metadaten hätte diese Vorgehensweise sicherlich Vorteile, gleichzeitig würden jedoch neue Probleme bei der Erfassung digitaler Objekte und ihrer Metadaten entstehen. In den meisten Repositorien können Nutzer neue Objekte zur Veröffentlichung einreichen, wobei sie auch die zugehörigen Metadaten erfassen. Durch die zusätzliche Forderung der Eindeutigkeit würde das erschwert. Als ein erster Schritt in diese Richtung könnten Metadaten zusätzlich zur herkömmlichen Form auch als URIs aufgenommen werden. Bestrebungen zur Vermeidung oder Auflösung von Mehrdeutigkeiten in Metadaten wie zum Beispiel das Projekt ORCID zur Autorenidentifikation könnten Rückwirkungen auf die Konvertierung der Daten in RDF haben.

Ein weiteres interessantes Thema sind die Auswirkungen auf den Informationsaustausch mit Repositorien. In der vorliegenden Arbeit wird die Frage aufgeworfen, ob die OAI-PMH-Schnittstelle durch RDF und einen SPARQL-Endpoint abgelöst werden kann. Die bisherige Bereitstellung der Repositorieninhalte hat den Nachteil, dass OAI-PMH nur im Repositorienumfeld verwandt wird, während RDF einen generischen Zugang bietet.

All diese Aspekte bieten einen guten Ausgangspunkt für die weitere Entwicklung von Softwarelösungen für Repositorien. Letztlich spiegeln sie Punkte der allgemeinen Diskussion über das Semantic Web und die Nutzung von Linked Data wider.

Literatur

Aschenbrenner, Andreas und Heike Neuroth (2011). *Forschungsdaten-Repositorien*. In: Handbuch Forschungsdatenmanagement. Hrsg. von Stephan Büttner, Hans-Christoph Hobohm und Lars Müller. Bad Honnef: Bock und Herchen. ISBN: 978-3-88347-283-6. URN: urn:nbn:de:kobv:525-opus-2328.

Becker, Pascal-Nicolas und Fabian M. Fürste (2013). *Sollen wir Bibliothekare jetzt alle Informatiker werden?* In: BuB – Forum Bibliothek und Information 65.7/8, S. 512–514. ISSN: 1869-1137.

Berners-Lee, Tim (1998). *Cool URIs don't change*. URL: http://www.w3.org/Provider/Style/URI (abgerufen am 06. 06. 2014).

Berners-Lee, Tim (2006). *Linked Data*. Design Issues. URL: http://www.w3.org/DesignIssues/LinkedData.html (abgerufen am 06. 06. 2014).

Berners-Lee, Tim und Mark Fischetti (1999). *Weaving the Web. The Past, Present and Future of the World Wide Web by its Inventor*. London: Orion Bunsiness Books. ISBN: 0-7528-2090-7.

Berners-Lee, Tim, James Hendler und Ora Lassila (2001). *The Semantic Web*. In: Scientific american 284.5, S. 28–37.

Bizer, Christian, Richard Cyganiak und Tom Heath (2007). *How to Publish Linked Data on the Web*. URL: http://wifo5-03.informatik.uni-mannheim.de/bizer/pub/LinkedDataTutorial/ (abgerufen am 06. 06. 2014).

Bizer, Christian, Tom Heath und Tim Berners-Lee (2009). *Linked Data – The Story So Far*. In: International Journal on Semantic Web and Information Systems (IJSWIS) 5.3, S. 1–22. DOI: 10.4018/jswis.2009081901.

Consultative Commitee for Space Data Systems (2012). *Reference model for an Open Archival Information System (OAIS)*. Recommended Practice, CCSDS 650.0-M-2, Magenta Book. Washington, DC, USA. URL: http://public.ccsds.org/publications/archive/650x0m2.pdf (abgerufen am 06. 06. 2014).

Deutsche Initiative für Netzwerkinformationen e.V. – Arbeitsgruppe Elektronisches Publizieren (2011). *DINI-Zertifikat Dokumenten- und Publikationsservice 2010*. Göttingen. URN: urn:nbn:de:kobv:11-100182794.

Deutsche Nationalbibliothek (2012). *Lieferung von Metadaten für Netzpublikationen an die Deutsche Nationalbibliothek. Metadaten-Kernset im Format XMetaDiss-Plus Version 2.2 (OAI-Schnittstelle)*. Version 1.1. Frankfurt a.M. URN: urn:nbn:de:101-2012022237.

DuCharme, Bob (2013). *Learning SPARQL*. 2nd Edition. Beijing: O'Reilly Media. ISBN: 978-1-449-37143-2.

Ginsparg, Paul (1994). *First steps towards electronic research communication*. In: Computers in Physics 8.4, S. 390–396. ISSN: 0894-1866.

Heath, Tom und Christian Bizer (2011). *Linked Data: Evolving the Web Into a Global Data Space*. Synthesis Lectures on Web Engineering Series. Morgan & Claypool. ISBN: 978-1-60845-431-0. DOI: 10.2200/S00334ED1V01Y201102WBE001.

Hinze, Annika, Ralf Heese, Markus Luczak-Rösch und Adrian Paschke (2012). *Semantic Enrichment by Non-experts: Usability of Manual Annotation Tools*. In: The Semantic Web – ISWC 2012. Hrsg. von Philippe Cudré-Mauroux u. a. Bd. 7649. Lecture Notes in Computer Science. Berlin: Springer, S. 165–181. ISBN: 978-3-642-35175-4. DOI: 10.1007/978-3-642-35176-1_11.

Hitzler, Pascal, Markus Krötzsch, Sebastian Rudolph und York Sure (2008). *Semantic Web: Grundlagen*. Berlin: Springer. ISBN: 978-3-540-33993-9.

nestor – Kompetenznetzwerk Langzeitarchivierung und Langzeitverfügbarkeit Digitaler Ressourcen für Deutschland, Hrsg. (2013). *Referenzmodell für ein Offenes Archiv-Informations-System – Deutsche Übersetzung 2.0*. nestor-materialien 16. Frankfurt a.M. URN: urn:nbn:de:0008-2013082706.

Open Society Institute (2004). *A Guide to Institutional Repository Software*. 3rd Edition. New York. URL: http://www.soros.org/openaccess/pdf/OSI_Guide_to_IR_Software_v3.pdf (abgerufen am 06. 06. 2014).

Upmeier, Arne (2008). *Digitale Repositorien an mittleren Universitätsbibliotheken. Anforderungen und Möglichkeiten am Beispiel des Dokumentenservers GEB des Bibliothekssystems Gießen*. Masterarbeit. Humboldt Universität zu Berlin. URL: http://geb.uni-giessen.de/geb/volltexte/2008/6719/pdf/UpmeierArne-2008-09-29.pdf (abgerufen am 06. 06. 2014).

Vierkant, Paul u. a. (2012). *2012 Census of Open Access Repositories in Germany*. Poster präsentiert auf den Open Access Tagen 2012 Wien. URN: urn:nbn:de:kobv:11-100204211.

W3C Technical Architecture Group (TAG) (2006). *URNs, Namespaces and Registries*. Draft TAG Finding. URL: http://www.w3.org/2001/tag/doc/URNsAndRegistries-50-2006-08-17 (abgerufen am 06. 06. 2014).

Eidesstattliche Erklärung

Ich versichere hiermit an Eides statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Quellen und Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind in dieser Arbeit angegeben. Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Berlin, den 17. Juni 2014

(Pascal-Nicolas Becker)