Konzeption eines RDF-Vokabulars

für die Darstellung von

COUNTER-Nutzungsstatistiken

innerhalb des

Electronic Resource Management Systems

der Universitätsbibliothek Leipzig

Masterarbeit

an der

Hochschule für Technik, Wirtschaft und Kultur Leipzig

Fakultät Medien

Studiengang Bibliotheks- und Informationswissenschaft

vorgelegt von

Annika Domin

Leipzig 2014

Domin, Annika:

Konzeption eines RDF-Vokabulars für die Darstellung von COUNTER-Nutzungsstatistiken innerhalb des Electronic Resource Management Systems der Universitätsbibliothek Leipzig / Annika Domin. - 2014. - 101, XXXVI Bl. : Ill., graph. Darst. + Kt.-Beil., 1 CD-ROM
Leipzig, Hochschule für Technik, Wirtschaft und Kultur, Masterarbeit, 2014

Betreuer:	Prof. Dr. Robert Müller, HTWK Leipzig, Fakultät Medien
Zweitgutachter:	Prof. Dr. Michael Frank, HTWK Leipzig, Fakultät Informatik, Mathematik und Naturwissenschaften

Erstellt in Kooperation mit der Universitätsbibliothek Leipzig.

Die vorliegende Masterarbeit dokumentiert die Erstellung eines RDF-basierten Vokabulars zur Darstellung von Nutzungsstatistiken elektronischer Ressourcen, die nach dem COUNTER-Standard erstellt wurden. Die konkrete Anwendung dieses Vokabulars bildet das *Electronic Resource Management System* (ERMS), welches momentan von der *Universitätsbibliothek Leipzig* im Rahmen des kooperativen Projektes AMSL entwickelt wird. Dieses basiert auf Linked Data, soll die veränderten Verwaltungsprozesse elektronischer Ressourcen abbilden können und gleichzeitig anbieterunabhängig und flexibel sein. Das COUNTER-Vokabular soll aber auch über diese Anwendung hinaus einsetzbar sein.

Die Arbeit gliedert sich in die beiden Teile *Grundlagen* und *Modellierung*. Im ersten Teil wird zunächst die bibliothekarische Notwendigkeit von ERM-Systemen herausgestellt und der Fokus der Betrachtung auf das Teilgebiet der Nutzungsstatistiken und die COUNTER-Standardisierung gelenkt. Anschließend werden die technischen Grundlagen der Modellierung betrachtet, um die Arbeit auch für nicht mit Linked Data vertraute Leser verständlich zu machen.

Darauf folgt der Modellierungsteil, der mit einer Anforderungsanalyse sowie der Analyse des den COUNTER-Dateien zugrunde liegenden XML-Schemas beginnt. Daran schließt sich die Modellierung des Vokabulars mit Hilfe von RDFS und OWL an. Aufbauend auf angestellten Überlegungen zur Übertragung von XML-Statistiken nach RDF und der Vergabe von URIs werden anschließend reale Beispieldateien manuell konvertiert und in einem kurzen Test erfolgreich überprüft. Den Abschluss bilden ein Fazit der Arbeit sowie ein Ausblick auf das weitere Verfahren mit den Ergebnissen.

ECCE HUBERTUS, SCIURUS MAGNIFICUS, INGENIOSUS ARTIFICIOSUSQUE.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AKSW.....Agile Knowledge Engineering and Semantic Web

CoP 4 .....Code of Practice for e-Resources, Release 4

CoP .....Code of Practice

COUNTER.....Counting Online Usage of NeTworked Electronic Resources

DLF .....Digital Library Federation

DNB .....Deutsche Nationalbibliothek

DTD .....Dokumenttyp-Definition

EFRE .....Europäischen Fonds für regionale Entwicklung

ELAG .....European Library Automation Group

ERMI .....Electronic Resource Management Initiative

ERMS .....Electronic Resource Management System

IETF .....Internet Engineering Task Force

IRI .....Internationalized Resource Identifier

NISO .....National Information Standards Organization

OWL.....Web Ontology Language OWL

Qname.....Qualified Name.

RDF.....Resource Description Framework

RDFS .....RDF Schema

RFC .....Request for Comments, Spezifikation der IETF

SGML.....Standard Generalized Markup Language

SLUB .....Sächsischen Landesbibliothek – Staats- und Universitätsbibliothek Dresden

SPARQL.....SPARQL Protocol And RDF Query Language

SUSHI .....Standardized Usage Harvesting Initiative

Turtle.....Terse RDF Triple Language

UB Leipzig .....Universitätsbibliothek Leipzig

URI.....Uniform Resource Identifier

W3C.....World Wide Web Consortium

Web, WWW.....World Wide Web

XML .....eXtensible Markup Language

1 Einleitung

1.1 Problematik, Ziel und Abgrenzung

Im Zuge der Etablierung elektronischer Ressourcen in Bibliotheksbeständen fallen mit den neuen, digitalen Medienarten gleichermaßen neue Verwaltungsvorgänge an. Anstatt herkömmliche Printausgaben einmalig zu erwerben, werden Verträge über die Nutzung von Lizenzpaketen abgeschlossen – Lizenzierungsvorgänge, die in einem Bibliotheksmanagementsystem erfassbar sein müssen, es in herkömmlichen bibliothekarischen Anwendungen allerdings nicht sind.

Bislang erfolgte die Verwaltung der elektronischen Ressourcen an der *Universitätsbibliothek Leipzig* (UB Leipzig) größtenteils manuell in einzelnen Dateien, was nicht die für die erfolgreiche Verwaltung von E-Ressourcen nötigen Möglichkeiten an vernetzter Betrachtung bot. Da bisher auf dem Markt befindliche proprietäre *Electronic Resource Management Systems* (ERMS) zwar mit den vorhandenen Bibliothekssystemen kompatibel sind, aber meist wenig Gestaltungsfreiraum bieten und stets eine Abhängigkeit von Softwareanbietern bedeuten, hat sich die UB Leipzig dazu entschlossen, im Rahmen eines Projektes ein eigenes System zu entwickeln. In ein ERMS fließen sehr unterschiedliche Daten aus verschiedensten Quellen ein, die miteinander verknüpft und später zumindest in Teilen auch wieder an Kooperationspartner weitergegeben werden müssen. Daher soll das geplante System aufgrund der großen Flexibilität auf Linked-Data-Technologie basieren. Alle Daten sollen entsprechend basierend auf RDF, dem im Linked-Data-Bereich vorherrschenden *Resource Description Framework*, verwaltet werden. Die Entwicklung des ERMS geschieht im Teilprojekt *AMSL - Linked-Data-basiertes Electronic Resource Management* im Rahmen des Projektes *Datenmanagement und ERM* gemeinsam mit der *Sächsischen Landesbibliothek – Staats- und Universitätsbibliothek Dresden* (SLUB) und weiteren Partnern.

Einen Bereich des Managements elektronischer Ressourcen, der in das geplante ERMS zu integrieren ist, stellen die Nutzungsstatistiken dar, bei denen im Vergleich zu konventionellen Medien neue Problematiken auftreten. Die lizenzierten elektronischen Medien werden überwiegend von den Anbietern auf ihren eigenen Servern zur Benutzung vorgehalten, die lizenznehmenden Bibliotheken besitzen keinen direkten Zugriff auf diese Server. Entsprechend fehlt ihnen die Möglichkeit, selbst

Statistiken über die Nutzung der Medien zu erstellen. Deshalb übernehmen die Medienanbieter diese Aufgabe und stellen ihren Kunden die Statistiken zur Verfügung. Der Vereinheitlichung und Standardisierung dieser Nutzungsberichte widmet sich seit Anfang der 2000er Jahre das *Projekt COUNTER*, das Anbieter dazu anhält, ihre Berichte entsprechend der erarbeiteten Report-Schemata zu erstellen. Durch die Integration solcher standardisierter Nutzungsstatistiken in das geplante ERMS soll an der UB Leipzig die bisher nicht ausreichend gegebene Übersichtlichkeit und Vergleichbarkeit dieser Daten hergestellt werden, um eine fundierte Bewertung der Nutzung einzelner elektronischer Ressourcen zu ermöglichen. Für diese Integration ist es erforderlich, dass die Nutzungsdaten dem restlichen System entsprechend im Linked-Data-Format vorliegen.

Ziel der vorliegenden Masterarbeit ist es daher, im Rahmen des AMSL-Projektes ein RDF-Vokabular zu erstellen, das die Grundlage bietet, alle in COUNTER-konformen Reports enthaltenen Informationen zu erfassen und miteinander zu verknüpfen, sodass eine RDF-basierte Verarbeitung von Nutzungsstatistiken innerhalb des ERMS der UB Leipzig ermöglicht wird. Der Vorgang der Modellierung soll dabei ausführlich dokumentiert werden, um zum einen die Arbeitsergebnisse für die sie implementierenden Projektmitarbeiter nachvollziehbarer zu machen und um zum anderen weiteren an der Problematik interessierten Lesern Ansätze und Ideen für eigene Projekte auf den Gebieten des ERM und der Linked Data zu bieten. Um auch für potentielle andere Anwender einen Nutzen über die konkrete Implementierung in der UB Leipzig hinaus zu haben, soll das Vokabular nicht zu eng an diesen Einsatz angepasst werden, sondern eine möglichst genaue Umsetzung des standardisierten COUNTER-Schemas darstellen. Da COUNTER-konforme Anbieter seit dem Release 4 verpflichtet sind, ihre Berichte auch im XML-Format zur Verfügung zu stellen, orientiert sich das zu erstellende Vokabular an dem zugrunde liegenden XML-Schema.

Die im XML-Format empfangenen Daten müssen in der Praxis automatisiert in das RDF-Format übertragen werden. Die Aufgabe der Datenkonvertierung wird in dieser Masterarbeit durch die an das XML-Schema angelehnte Konzeption des Vokabulars zwar berücksichtigt und durch die Vorbereitung einer XML-RDF-Konkordanz unterstützt, jedoch nicht gelöst. Auch die RDF-Abfragesprache SPARQL (*SPARQL Protocol And RDF Query Language*), die später neben anderen Funktionalitäten der Basissoftware *OntoWiki* zur Datenabfrage genutzt werden wird, ist nicht Teil dieser Arbeit, da für sinnvolle Queries das noch nicht fertiggestellte Gesamtsystem einbezogen werden müsste.

1.2 Zielgruppe, Methodik und Aufbau

Die Art der Aufbereitung und Darstellung der Thematik orientiert sich an der Zielgruppe dieser Masterarbeit, die sich insbesondere an bibliothekarisches Fachpersonal und an mit Bibliotheken assoziierte IT-Fachkräfte richtet. Bei beiden Teilen dieser Zielgruppe wird ein gewisses informationstechnisches Basiswissen vorausgesetzt, sodass grundlegende Konzepte wie beispielsweise die *eXtensible Markup Language* XML nicht mehr ausführlich erläutert, sondern lediglich kurz aufgegriffen werden. Es sollen sowohl solche Leser angesprochen werden, die konkret in ihrer Institution an der Frage der Verwaltung von Nutzungsstatistiken elektronischer Ressourcen arbeiten, als auch Leser, die sich allgemein für den bibliotheksspezifischen Einsatz der Linked-Data-Technologie interessieren.

Die vorliegende Arbeit zeigt einen solchen praktischen Einsatz von Linked Data im Bibliotheksumfeld auf. Sie dokumentiert den Prozess, wie nach gemeinsamen Grundüberlegungen und unter Berücksichtigung gegebener Standards ein Modell entwickelt werden kann, das den Bedürfnissen der Bibliothek gerecht wird. Der Schwerpunkt der Arbeit liegt auf der praktischen Entwicklung des Vokabulars, wobei das Vorgehen in seinen einzelnen Schritten erläutert und begründet wird. Das Ergebnis ist eine Variante von diversen möglichen Formen des Vokabulars, weshalb die einzelnen Entscheidungen, die zu den gewählten Lösungen geführt haben, jeweils diskutiert und verständlich gemacht werden. Da die softwareseitige Implementierung des Vokabulars erst nach Beendigung dieser Arbeit geleistet wird, kann das fertige Vokabular nicht unter realen Bedingungen getestet werden. Um trotzdem dessen Funktionalität überprüfen zu können, werden Beispieldatensätzen der UB Leipzig manuell in das RDF-Format umgewandelt und ebenfalls manuell in die später zu nutzende, noch nicht endgültig angepasste Software importiert. Dadurch können zumindest Fehler in der Umsetzung des XML-Schemas sowie in der grundlegenden Verknüpfung der RDF-Ressourcen untereinander erkannt werden.

Die Arbeit ist in die beiden Hauptteile *Grundlagen* und *Modellierung* aufgegliedert. Der erste Teil vermittelt das für das Verständnis der Arbeit notwendige Basiswissen. Für nicht direkt von der Nutzungsdatenverwaltung betroffene Leser wird dieser Themenkomplex zu Beginn kurz erläutert. Dabei wird zunächst allgemein in das Feld des Electronic Resource Managements eingeführt, um daraufhin die spezielle Thematik des Umgangs mit Nutzungsstatistiken elektronischer Ressourcen sowie das auf diesem Gebiet federführende Projekt COUNTER zu thematisieren. Daran schließt sich eine genauere Vorstellung des AMSL-Projektes an, welches die geschilderten Problematiken aufgreift.

Entsprechend der überwiegend bibliothekarischen Zielgruppe werden die technischen Grundlagen der Vokabularerstellung systematisch aufeinanderfolgend eingeführt und in ihrem grundlegenden Aufbau erläutert. Dies geschieht losgelöst von der Thematik der Nutzungsdaten anhand einfacher Beispiele und hat den Zweck, die vorliegende Arbeit auch zuvor nicht mit Linked Data vertrauten Lesern verständlich zu machen. Ein Lehrbuch kann und soll dabei aber keinesfalls ersetzt werden. Nach dem größtenteils als bekannt vorausgesetzten XML wird daraufhin das Konzept von Linked Data vorgestellt und am Beispiel des Semantic Webs verdeutlicht. In das Gebiet der Organisation semantisch verknüpfter Daten beginnt das darauffolgende Kapitel über Grundkonzepte der Modellierung einzuführen. Daran schließen sich nacheinander die Vorstellungen der immer komplexer werdenden Modellierungssprachen bzw. -konzepte RDF, RDFS und OWL an, die im Falle von RDF in den Grundzügen noch recht ausführlich, bei RDFS und OWL aber nur noch wesentlich oberflächlicher eingeführt werden.

Der zweite Teil der Arbeit widmet sich der konkreten Modellierung des Vokabulars. Dazu werden zunächst die notwendigen Vorarbeiten thematisiert, die mit einer Anforderungsanalyse beginnen, welche die grundsätzliche Richtung der Modellierung bestimmt. Darauf folgen die Analyse des Aufbaus des COUNTER-XML-Schemas anhand exemplarischer Beispiele sowie die auf diesen beiden Untersuchungen basierende Grundkonzeption des Vokabulars. Abschließend werden die für die Modellierung verwendeten Programme vorgestellt, um sowohl das Vorgehen genauer zu dokumentieren als auch eine Empfehlung für andere Modellierer auszusprechen.

Das nächste Kapitel widmet sich der eigentlichen Erstellung des RDF-Vokabulars. Dazu werden zunächst die grundlegenden Elemente in RDFS modelliert, um anschließend Erweiterungen mit Hilfe des semantisch mächtigeren OWL durchzuführen. Nach der Modellierung von Klassen, Eigenschaften und einigen Individuen erläutert das nächste Unterkapitel die Überlegungen zur Übertragung der Reportdaten von XML nach RDF. Dadurch soll zusammen mit der in Anhang C angefügten XML-RDF-Konkordanz eine Grundlage für die spätere Erstellung des XML-RDF-Konverters durch Projektmitarbeiter geschaffen werden. Die bei der Überführung der Reportdaten notwendige Identifikation der RDF-Ressourcen ist das Kernproblem des letzten Unterkapitels, welches die URI-Vergabe thematisiert. All diese Modellierungsschritte werden aufgrund ihres sich wiederholenden Aufbaus nicht für die Erstellung des gesamten Vokabulars dargestellt, sondern anhand exemplarischer Beispiele verdeutlicht.

Das nächste Kapitel beschreibt den an einer OntoWiki-Installation in der UB Leipzig durchgeführten, aufgrund des Arbeitsstandes am Gesamtsystem erst vorläufig angelegten Test des Vokabulars mitsamt dessen Konzeption, Durchführung und den Ergebnissen. Abschließend folgt ein Fazit der Masterarbeit sowie ein Ausblick auf die geplante weitere Verfahrensweise mit den Ergebnissen.

Dem Hauptteil der Arbeit folgen nach dem Literatur- und Quellenverzeichnis sowohl gedruckte als auch elektronische Anhänge. Anhang A umfasst nach den nur exemplarischen Beschreibungen im Hauptteil nun alle modellierten Klassen und Eigenschaften des RDF-Vokabulars mit kurzen Erklärungen zu deren Einsatz, um einen Überblick über das Gesamtvokabular zu geben. Anhang B fasst die Regelungen zur URI-Vergabe für alle zu erstellenden Ressourcen zusammen und enthält Begründungen zu deren Festlegung. Anhang C schließlich umfasst die erstellte XML-RDF-Konkordanz.

Die beiliegende CD-ROM beinhaltet den elektronischen Anhang D. Dieser umfasst Kopien aller verwendeten und erstellten Dateien sowie eine PDF-Version der gedruckten Textteile. Neben der CD-ROM liegt der Arbeit außerdem ein im A3-Format gedrucktes Schema des erstellten Vokabulars bei, das der Orientierung während des Lesens dienen soll.

1.3 Forschungsstand und Quellenlage

Die vorliegende Arbeit beschäftigt sich mit der Linked-Data-Repräsentation von Nutzungsstatistiken elektronischer Ressourcen im Zuge der Entwicklung eines ERMS. Dies stellt eine Vereinigung der Themenkomplexe des ERM und der Linked-Data-Technologien dar. Beides wurde in der Bibliothekswelt bisher eher getrennt betrachtet.

Zum Elektronischen Ressourcenmanagement existieren etliche Ansätze, vorangetrieben durch die Arbeit diverser bibliothekarischer Zusammenschlüsse und Projekte. Hier sind insbesondere die umfangreichen Untersuchungen der DLF ERMI, der *ERM-Initiative* der *Digital Library Federation*, zu nennen, die in Kapitel *2.1 Electronic Resource Management* angesprochen werden. Ergebnis dieser Untersuchungen war unter anderem ein detaillierter Anforderungskatalog für ERM-Systeme, der den Grundstein für entsprechende Neuentwicklungen legt. Auf dem Teilgebiet der Vereinheitlichung von Nutzungsdaten elektronischer Ressourcen ist das Projekt COUNTER federführend, des-

sen Standardisierungen in Datenanbieter- und Bibliothekskreisen etabliert sind. Die COUNTER-Daten werden bisher allerdings nur im Microsoft Excel-Tabellenformat sowie als XML-Dateien angeboten. Arbeiten zur Linked-Data-Repräsentation der COUNTER-Statistiken sind bisher nicht bekannt.

In Hinblick auf bibliothekarische Aktivitäten im Linked-Data-Bereich wurde bereits viel Arbeit geleistet. Es existieren etliche Projekte zur Repräsentation bibliothekarischer Daten als Linked Data, sowohl der verschiedenen nationalen und internationalen Bibliotheksorganisationen sowie auch anderer Akteure wie dem *World Wide Web Consortium* (W3C). Dieses initiierte beispielsweise von 2010 bis 2011 die *Library Linked Data Incubator Group*, welche die globale Interoperabilität von Bibliotheksdaten verbessern und Bibliotheken den Einstieg in das Semantic Web erleichtern wollte.[1] Ähnliche Ziele verfolgt das Unternehmen OCLC, das sowohl prägend auf die konzeptuelle Gestaltung des Einbringens von Bibliothekdaten in das Semantic Web einwirken möchte, als sich auch aktiv über seinen *WorldCat* an der Zurverfügungstellung solcher Daten beteiligt.[2] Die *Deutsche Nationalbibliothek* (DNB) hat ebenfalls einen Linked-Data-Service eingerichtet und möchte mit ihren intellektuell erstellten bibliographischen und Normdaten das Semantic Web qualitativ bereichern. So werden beispielsweise die Daten der GND als Linked Data angeboten,[3] unterstützt durch die GND Ontology.[4] Auch umgekehrt werden von anderen als Linked Open Data publizierte Informationen durch die DNB zur Anreicherung von Normdaten genutzt. So werden etwa die geographischen Normdaten seit Februar 2014 mit Koordinaten aus der freien Zusammenstellung *GeoNames*[5] ergänzt.[6] Die DNB beteiligt sich weiterhin an der internationalen *Bibliographic Framework Initiative* (BIBFRAME), die bibliographische Formate unter Berücksichtigung von Semantic Web-Technologien überarbeiten will.[7] Für die in den nächsten Jahren auch in Deutschland als neues Standardregelwerk für die Katalogisierung eingeführten RDA-Regeln (*Resource Description and Access*) existieren bereits umfangreiche Vokabulare zur Darstellung der bibliographischen Daten im Linked-Data-Format.[8]

[1] Vgl. Library Linked Data Incubator Group [Elektronische Ressource].
[2] Vgl. Data Strategy and Linked Data [Elektronische Ressource].
[3] Vgl. Linked Data Service der Deutschen Nationalbibliothek [Elektronische Ressource].
[4] Siehe unter http://d-nb.info/standards/elementset/gnd#.
[5] Siehe unter http://www.geonames.org/.
[6] Vgl. Linked Data Service der Deutschen Nationalbibliothek [Elektronische Ressource].
[7] Vgl. BIBFRAME - Bibliographic Framework Initiative [Elektronische Ressource].
[8] Siehe unter http://rdvocab.info/.

All diese bibliothekarischen Anstrengungen im Bereich der Linked Data zielen allerdings hauptsächlich auf die Repräsentation bibliographischer Daten ab. Eine Nutzung der Technologie für bibliotheksinterne Verwaltungsprozesse von Ressourcen ist derzeit nicht bekannt und, wie auch die Linked-Data-Umsetzung des COUNTER-Schemas, neuartig im AMSL-Projekt. Entsprechend existieren auch noch keine Publikationen, die das Problemfeld der vorliegenden Arbeit in seiner Gesamtheit betrachten.

Zu den bibliothekarischen Aspekten des ERM wurden insbesondere im US-amerikanischen Raum etliche Berichte und Zeitungsartikel veröffentlicht, die einen guten Themeneinstieg und Überblick bieten. Für genauere Informationen zu Aspekten und Spezifikationen zu planender ERM-Systeme ist der Abschlussbericht der DLF ERMI aus dem Jahr 2004 unerlässlich. Die für die Nutzungsstatistiken relevanten Festlegungen sind direkt auf den Webseiten des Projekts COUNTER einsehbar, wo auch alle diesbezüglichen Publikationen des Projekts frei abrufbar sind. Explizit auf die XML-Umsetzung bezogene Informationen sind auf den Webseiten der US-amerikanischen Standardisierungsorganisation NISO (*National Information Standards Organization*) ebenfalls frei zugänglich zusammengefasst.

Hinweise zu dem korrekten Aufbau einer Anforderungsanalyse und der Durchführung von Tests sind umfassend in der Fachliteratur der Informatik zu finden, dort allerdings für die Zwecke dieser Arbeit wesentlich zu tief und detailliert erörtert, da hier beides nur in Anlehnung an die gängigen Methoden durchgeführt werden kann. Als eine gute Alternative haben sich im Internet verfügbare Präsentationen zu Software-Engineering-Vorlesungen verschiedener Universitäten aus den letzten Jahren herausgestellt. Diese sind relativ knapp auf die essentiellen Bestandteile reduziert und daher für den Rahmen dieser Arbeit wesentlich angemessener als die ausführlichen Fachpublikationen.

Da ein Großteil der Arbeit aus der selbstständigen Modellierung besteht, waren insbesondere leicht verständliche Lehrbücher für die Konzepte von Linked Data und Ontologien sowie den grundlegenden Aufbau von RDF, RDFS und OWL notwendig. Deutschsprachige Literatur, im Speziellen solche, die auch für Fachfremde verständlich ist, findet sich dabei recht selten. Eines dieser Werke ist die Aufsatzsammlung *Semantic Web : Wege zur vernetzten Wissensgesellschaft*, herausgegeben 2006 von Tassilo Pelligrini bei Springer, welche ein breites Spektrum an einführenden Beiträgen zu diversen Aspekten des Semantic Webs bietet. Obwohl etliche Aufsätze thematisch zu weit von den Zielen dieser Arbeit abweichen, sind andere doch für den allgemeinen Einstieg in das Themenfeld der Linked Data gut geeignet. Monothematisch und dafür fachlich tiefergehend angelegt ist das 2011 veröffentlichte Werk *Ontologien* von Heiner Stuckenschmidt aus der Reihe *Informatik im Fokus*, das

sich ausführlich mit Ontologien und den dahinterliegenden Konzepten beschäftigt. Dieses Werk ist als Hintergrundlektüre sehr zu empfehlen, für die praktische Verwendung zum Modellieren aber aufgrund der abstrakten Herangehensweise nicht zielführend.

Für diese Zwecke eignet sich dagegen das englischsprachige Lehrbuch *Semantic Web for the Working Ontologist* hervorragend, das in zweiter, stark überarbeiteter Auflage im Jahr 2011 von Dean Allemang und Jim Hendler bei Morgan Kaufmann veröffentlicht wurde. Jim Hendler war neben Berners-Lee einer der drei ursprünglichen Initiatoren des Semantic Web-Gedankens,[9] was aber keineswegs einen zu hohen Schwierigkeitsgrad des genannten Lehrbuches bedeutet. Es richtet sich im Gegenteil explizit an einen nicht aus der Informatik stammenden Leserkreis und führt anhand etlicher Beispiele und in leicht verständlicher Sprache in die Konzepte von Linked Data und in die praktische Anwendung von Modellierungssprachen ein. Bei tiefer greifenden Fragen zu RDF war auch das von Shelly Powers bereits im Jahr 2003 verfasste Lehrbuch *Practical RDF* aus dem Verlag O'Reilly hilfreich, das allerdings in einigen Teilen bereits zu stark veraltet war.

Für konkrete syntaktische Probleme und exakte Definitionen sind die Webseiten des W3C und der IETF (*Internet Engineering Task Force*) unerlässlich. Die von den Standardisierungsorganisationen herausgegebenen Spezifikationen sind zwar häufig für Laien eher schwer verständlich, werden allerdings in vielen Fällen durch speziell für Einsteiger verfasste Dokumente mit weniger formeller Sprache und erläuternden Beispielen ergänzt.

[9] Vgl. Web 3.0 [Interview with Jim Hendler] [Elektronische Ressource].

TEIL I - Grundlagen

2 Bibliothekarische Ausgangssituation

2.1 Electronic Resource Management

Elektronische Ressourcen sind inzwischen ein etablierter Teilbestand in Bibliotheken. Ihre Verwaltung allerdings gestaltet sich in etlichen Punkten deutlich abweichend und oftmals vielschichtiger als die der konventionellen, gedruckten Medien.

Während ein gedrucktes Werk einmal gekauft und damit dauerhaft in den Bestand übernommen wird, sind elektronische Ressourcen dynamisch. Es werden nicht die physischen Medien selbst erworben, sondern lediglich Nutzungsrechte an den Werken lizenziert. Derartige Lizenzen können auslaufen, die Lizenzgeber und -nehmer können wechseln. Auch Pakete von Lizenzen können sich in ihrer Zusammensetzung verändern.[10] Solche verbreiteten Lizenzpakete, die meist anstelle von Nutzungsrechten an einzelnen Medien erworben werden müssen, können mehrere Hundert Einheiten umfassen. Diese schiere Menge ist ein weiteres Problem, da eine konventionelle Katalogisierung auf der Ebene einzelner selbstständiger bibliographischer Einheiten ökonomisch nicht mehr angemessen scheint.[11] Neben der Vielzahl an Medien existiert auch eine Vielzahl an Bezugs-, Geschäfts- und Verlagsmodellen. Zeitschriftenabonnements können beispielsweise als sogenannte Parallel-Abonnements sowohl gedruckte Werke als auch digitale Nutzungsrechte umfassen, dabei sind Aufpreisen möglich, jedoch nicht immer der Fall. Nur digitale Rechte werden bei einem E-only-Abonnement erworben, das günstiger als das Parallel-Abonnement sein kann, aber nicht muss. Bibliotheken können sich für den Erwerb von Lizenzrechten auch zu Konsortien zusammenschließen, als die weitreichendste Variante eines solchen institutionsübergreifenden Erwerbes sind die Nationallizenzen zu nennen. Neben den diversen Arten des Lizenzerwerbs existieren ebenso viele Arten von Nutzungsrechten. Entsprechend ist es nötig, den Zugang zu den lizenzierten Dateien individuell zu beschränken, was eine komplexe Nutzerverwaltung nötig macht.[12]

[10] Vgl. Kowalak, Mario: Electronic Resource Management (ERM) in Bibliotheken [Elektronische Ressource], Folie 5.
[11] Vgl. ebd., Folie 6.
[12] Vgl. ebd., Folie 5.

Für diese Vielfalt neuartiger Aufgaben sind konventionelle *integrierte Bibliothekssysteme* (engl. *integrated library systems*, ILS) nicht ausgelegt. Neben der für gedruckte Medien schlicht nicht vorgesehenen Lizenzverwaltung ist es problematisch, dass die Verwaltung der elektronischen Ressourcen verschiedene Bereiche des bibliothekarischen Geschäftsganges einbindet, was zu separater oder doppelter Datenhaltung führen kann.[13] Auch die momentan als Übergangslösung in Bibliotheken wie der UB Leipzig manuell geführten Listen sind über verschiedene Bearbeiter verteilt, was neben ihrer fehlenden Möglichkeit zur Darstellung von Zusammenhängen ein weiteres großes Problem ist. Ziel eines ERMS ist es daher auch, Daten möglichst an nur einer Stelle vorzuhalten.[14]

Ein ERMS soll eine einheitliche Verwaltung elektronischer Ressourcen ermöglichen. Um vereinzelte lokale Ad-hoc-Lösungen zu umgehen, wurde von der *Digital Library Federation* (DLF) Die *Electronic Resource Management Initiative* (ERMI) ins Leben gerufen. Die aus 33 institutionellen Mitgliedern, meist Hochschulbibliotheken, und fünf verbundenen Organisationen bestehende US-amerikanische DLF strebt generell an, Bibliotheksbestände und -dienstleistungen durch das Vorantreiben des Einsatzes elektronischer Informationstechnologie zu verbessern. Im Falle von ERMI sollten konkrete Probleme in der Verwaltung elektronischer Ressourcen analysiert, koordinierte Lösungsstrategien entwickelt und diese an die Bibliotheksöffentlichkeit herangetragen werden. Das Ergebnis ist ein umfassender Abschlussbericht aus dem Jahre 2004, dem unter anderem ein Anforderungskatalog an ERM-Systeme beiliegt.[15]

Die Vision der ERMI ist ein ERM-System, welches „das Management von Informationen und Arbeitsprozessen zur effizienten Auswahl, Evaluierung, Erwerbung und Pflege unterstützt und den Zugang zu den E-Ressourcen in Einklang mit deren Vertrags- und Lizenzbedingungen ermöglicht.“[16] Wichtig ist, dabei gängige Standards zu nutzen, um mit anderen Systemen kompatibel zu bleiben,[17] da das ERMS als solches nur als „eine Etappe auf dem Weg zum integrierten und einheitlichen Ressourcenmanagement“[18] zu sehen ist.

Die Vorteile des ERMS und seiner Bündelung von Informationen zeigt sich in nahezu allen Bereichen des Umgangs mit elektronischen Ressourcen. Bei der Marktsichtung hilft der Überblick über bereits

[13] Vgl. Kowalak, Mario: Electronic Resource Management (ERM) in Bibliotheken [Elektronische Ressource], Folie 6.
[14] Vgl. ebd., Folie 8.
[15] Vgl. Forward to the Report of the DLF ERM Initiative [Elektronische Ressource].
[16] übersetzt aus dem Englischen nach: Electronic Resource Management : Report of the DLF ERM Initiative [Elektronische Ressource], S. 49.
[17] Vgl. Kowalak, Mario: Electronic Resource Management (ERM) in Bibliotheken [Elektronische Ressource], Folie 21.
[18] Ebd., Folie 18.

lizenzierte Medien und eventuelle Überschneidungen verschiedener Pakete. Kommt es zu Lizenzverhandlungen mit Verlagen, ist im ERMS beispielsweise ersichtlich, wie viele Produkte vom selben Anbieter bereits lizenziert und wo dadurch möglicherweise Rabatte verhandelbar sind. Die Lizenzierung und der entsprechende Vertrag lassen sich detailliert im ERMS abbilden und für spätere Verhandlungen nutzen. Die Verwaltung von Nutzerdaten und -zugängen kann direkt an die Lizenzverwaltung gekoppelt werden, wodurch die Transparenz wesentlich erleichtert wird. Durch verknüpfte Nutzungsstatistiken, Problem- und Störfallberichte sowie die anfallenden Kosten lassen sich Kosten-Nutzen-Analysen erstellen, die für die Erwerbungspolitik große Bedeutung haben.[19]

Bisher auf dem Markt befindliche Systeme zur Verwaltung von Bibliotheksdaten lassen sich grob in drei Kategorien unterscheiden. Zunächst sind die herkömmlichen ILS zu nennen, die ausschließlich die auf konventionelle Ressourcen ausgerichteten Arbeitsprozesse unterstützen. Sie sind somit für das ERM nicht anwendbar. Daneben existieren sowohl reine ERM-Systeme, die nur für die Verwaltung elektronischer Ressourcen ausgelegt sind, als auch sogenannte *Next Generation Library Systems* oder *Unified Resource Management Systems*, die das Management von konventionellen und neuen Medien verbinden. Der Nachteil solcher Systeme liegt in ihrer überwiegend kommerziellen, proprietären Ausrichtung. Sie sind größtenteils Cloud-basierte *Software-as-a-Service*-Produkte, die zu einer starke Anbieterabhängigkeit führen. Durch ihre im Vergleich zu *Open-Source*-Anwendungen niedrigere Transparenz kann es beispielsweise erschweren, die Vollständigkeit der Implementierung von Standards einzuschätzen. Auch die Einbindung externer Daten ist überwiegend schwer bis unmöglich, eine individuelle Anpassung von Funktionalitäten an eigene Bedürfnisse meist nicht vorgesehen.[20] Es existieren also bereits softwareseitige Lösungen für die Probleme des ERM, diese erfüllen allerdings insbesondere bezüglich ihrer Flexibilität und Transparenz noch nicht alle Erwartungen der bibliothekarischen Anwender.

2.2 Nutzungsdaten elektronischer Ressourcen

Die angesprochenen Problemfelder in der Verwaltung elektronischer Ressourcen erweitern sich im Bereich der Nutzungsstatistiken um eine neue Schwierigkeit. Während konventionelle Medien direkt in der Bibliothek ausgeliehen oder konsumiert werden, werden die lizenzierten elektronischen

[19] Vgl. Kowalak, Mario: Electronic Resource Management (ERM) in Bibliotheken [Elektronische Ressource], Folie 25.
[20] Vgl. AMSL [Elektronische Ressource], S. 3.

Dateien im Allgemeinen auf Webservern der jeweiligen Anbieter vorgehalten. Wird ein Nutzer über eine durch die Bibliothek registrierte IP-Adresse oder ein entsprechendes Login vom Datenanbieter als nutzungsberechtigt eingestuft, wird ihm von diesem der Zugriff auf das lizenzierte Medium gestattet. Die Bibliothek ist also an der konkreten Nutzung eines Mediums nicht mehr direkt beteilig.

Das bedeutet, dass sie auch über keine direkte Möglichkeit mehr verfügt, diese Nutzung selbst statistisch zu messen. Wo bei konventionellen Medien die Ausleihen im eigenen System verzeichnet werden und Präsenznutzung über Beobachtung oder ähnliche selbst durchführbare Verfahren ebenfalls annähernd bestimmbar ist, sind Bibliotheken für die Nutzungsstatistiken ihrer lizenzierten elektronischen Medien auf eine Kooperation mit den Datenanbietern angewiesen. Diese sind dabei in der Lage, diverse verschiedene Aspekte der Nutzung zu erheben und etwa auch auf die Verfügbarkeit in unterschiedlichen Dateiformaten einzugehen, da beispielsweise dasselbe Journal sowohl als PDF als auch als HTML oder Postscript-Datei abrufbar sein kann. Dies muss bei der Erhebung von Nutzungsdaten berücksichtigt werden, was dazu beiträgt, dass diese im Vergleich zu herkömmlichen Nutzungsstatistiken deutlich komplexer werden können.

Damit Bibliotheken diese komplexen, von verschiedenen Anbietern erhaltenen Daten miteinander vergleichen und komfortabel auswerten können, ist eine Standardisierung der Nutzungsstatistiken sinnvoll. Dieser Aufgabe widmet sich das *Projekt COUNTER*, das seit 2003 *Codes of Practice* (CoP) genannte Standards für Nutzungsstatistiken elektronischer Ressourcen veröffentlicht. Das Akronym COUNTER steht für *Counting Online Usage of NeTworked Electronic Resources* und benennt damit das gewählte Aufgabengebiet des Projektes, das im Jahr 2002 als internationale Initiative zur Verbesserung der Verlässlichkeit von Online-Nutzungsstatistiken ins Leben gerufen wurde.[21] Das Projekt COUNTER ist örtlich in Großbritannien angesiedelt, wo es als eine *Not-for-Profit*-Organisation aufgestellt ist, die sich im Besitz ihrer Mitglieder befindet.[22] Die Mitglieder setzen sich aus industriellen Organisationen, Bibliotheks-Konsortien, einzelnen Bibliotheken, Verlegern und Datenanbietern zusammen.[23]

Das Ziel von COUNTER ist es, die Zuverlässigkeit, Kompatibilität und Konsistenz der von den Datenanbietern gelieferten Statistiken sicherzustellen.[24] Um dies zu erreichen, werden verschiedene Ele-

[21] Vgl. About COUNTER [Elektronische Ressource].
[22] Vgl. COUNTER FAQs [Elektronische Ressource].
[23] Vgl. COUNTER members [Elektronische Ressource].
[24] Vgl. COUNTER FAQs [Elektronische Ressource].

mente und der Aufbau der als *Reports* bezeichneten, zu liefernden statistischen Berichte reglementiert. Die dazu veröffentlichten Codes of Practice definieren einen Satz von Kern-Reports, welche jeweils andere Arten von Ressourcen und Statistiken abdecken und für die stets Inhalt, Format, Übertragungsmechanismen und Datenverarbeitungsregeln spezifiziert werden.[25]

Die aktuellste Version stellt der *Code of Practice for e-Resources, Release 4* (CoP 4) dar, der im April 2012 veröffentlicht wurde und seit Jahresbeginn 2014 von COUNTER-konformen Datenanbietern verbindlich umzusetzen ist. Er fasst erstmals Reports sowohl über Zeitschriften und Datenbanken als auch über Bücher und Nachschlagewerke zusammen, die zuvor in separaten CoPs geführt wurden, und integriert außerdem Reports über multimediale Inhalte.[26] Daneben existieren zwei weitere Codes of Practice. Der erste standardisiert Reports für Artikel, der zweite definiert den sogenannten *Usage Factor*,[27] der eine nutzungsbasierte Ergänzung zu dem etablierten, zitationsbasierten *Journal Impact Factor* darstellen soll.[28]

Datenanbieter, die ihre statistischen Daten konform zu den COUNTER-CoPs aufbauen, können sich nach einem Audit[29] als *COUNTER compliant vendor* registrieren lassen. Danach werden sie zum einen in den auf der Website des Projektes veröffentlichten Registern konformer Anbieter[30] verzeichnet und sind zum anderen berechtigt, ein entsprechendes Logo zu führen. Es werden momentan drei unabhängige Register für die drei verschiedenen CoPs geführt, um die beiden weniger verbreiteten CoPs für Artikel bzw. den Usage Factor nicht für alle konformen Datenbieter zur Verpflichtung zu erheben.[31]

Abb. 1
Logo "COUNTER compliant".

Neben der eigentlichen Standardisierung von Nutzungsstatistiken betreibt das Projekt COUNTER auch kooperative Forschung und hat so beispielsweise bereits den Einfluss von Verlagsplattformen auf die Nutzung untersucht. Eine solche Kooperation fand auch mit der US-amerikanischen Standardisierungsorganisation NISO statt und führte zu dem Protokoll SUSHI (*Standardized Usage Har-*

[25] Vgl. COUNTER FAQs [Elektronische Ressource].
[26] Vgl. The COUNTER Code of Practice for e-Resources: Release 4 [Elektronische Ressource], S. 1.
[27] Vgl. COUNTER FAQs [Elektronische Ressource].
[28] Vgl. Usage factor [Elektronische Ressource].
[29] Vgl. COUNTER register of vendors [Elektronische Ressource].
[30] Erreichbar unter http://projectcounter.org/compliantvendors.html.
[31] Vgl. COUNTER FAQs [Elektronische Ressource].

vesting Initiative), das ein automatisiertes Harvesting der Reports von verschiedenen Datenanbietern ermöglicht.[32] Im Zuge von SUSHI wurde außerdem ein XML-Schema für COUNTER-Reports entworfen, welches seit dem CoP 4 verbindlich umzusetzen ist.[33] Dieses XML-Schema ermöglicht die XML-Darstellung aller im CoP 4 vorgesehenen Reports. Es wird außerdem durch das SUSHI Komitee stetig weitergepflegt,[34] um die Deckungsgleichheit mit den bisher ungefähr alle zwei Jahre erscheinenden, aktuellen Tendenzen Rechnung tragenden Codes of Practice[35] zu gewährleisten.

2.3 Projekt AMSL

Das Projekt *AMSL* widmet sich der Erstellung eines unabhängigen, flexiblen und auf Linked-Data-Technologien basierenden ERM-Systems. Die Wahl von Linked Data trägt einerseits den schnellen Änderungen auf dem Markt elektronischer Ressourcen Rechnung, da das dem System zugrunde liegende Datenmodell schnell und leicht angepasst werden kann, während die eigentliche Software unverändert bleibt. Andererseits stellt Linked Data auch eine optimale Herangehensweise an eine der Hauptschwierigkeiten des ERM dar, die nicht in einer hohen Komplexität einzelner Arbeitsvorgänge, sondern in der Vielfalt an Datenquellen und -formaten und der starker Verknüpfung dieser Daten untereinander liegt. Über Linked Data lassen sich Fremddaten einfach einbinden, insbesondere da immer mehr bibliothekarische Daten in entsprechenden Formaten zur Verfügung gestellt werden. Nach der Integration in das System sind Linked-Data-Technologien ebenfalls sehr gut zur Verknüpfung der importierten Daten geeignet, um deren komplexe Beziehungen angemessen abbilden zu können.[36] Im Zuge der kooperativen Zusammenarbeit der sächsischen Hochschulbibliotheken ermöglicht Linked Data zudem einen einfachen Datenaustausch.

AMSL ist Teil des kooperativen Gesamtprojektes *Wissenschaftskommunikation im Semantischen Web*, an dem neben der UB Leipzig und der SLUB für die informationstechnische Fachkompetenz auch die Forschungsgruppe *Agile Knowledge Engineering and Semantic Web* (AKSW) am Leipziger *Institut für Angewandte Informatik* sowie das Unternehmen *Avantgarde Labs* beteiligt sind.[37] Das gesamte Projekt wird aus dem *Europäischen Fonds für regionale Entwicklung* (EFRE) und durch den

[32] Vgl. About COUNTER [Elektronische Ressource].
[33] Vgl. The COUNTER Code of Practice for e-Resources: Release 4 [Elektronische Ressource], S. 4.
[34] Vgl. SUSHI/COUNTER Schemas [Elektronische Ressource].
[35] Vgl. COUNTER FAQs [Elektronische Ressource].
[36] Vgl. AMSL [Elektronische Ressource], S. 4.
[37] Vgl. Projekt Datenmanagement und ERM [Elektronische Ressource].

Freistaat Sachsen finanziert[38] und läuft von Juni 2013 bis September 2014.[39] Die SLUB und Avantgarde Labs verfolgen das Teilprojekt *Entwicklung einer Datenmanagement-Plattform zur automatischen Verknüpfung von Bibliotheksdaten*. Im Zuge dessen wurde Ende Mai das Alpha-Release der Datenmanagementsoftware *d:swarm* veröffentlicht,[40] die ein Werkzeug zur Modellierung und Integration bibliographische Daten darstellt. d:swarm fügt Daten aus heterogenen Quellen in ein flexibles, graphenbasiertes Datenmodell ein und bindet diese in den Linked Open Data-Webgraphen ein.[41]

Währenddessen sind für das zweite Teilprojekt die UB Leipzig und AKSW zuständig. Entsprechend der oben geschilderten inhaltlichen Ausrichtung lautet der ausführliche Projektname *Entwicklung eines Electronic Resource Management Systems für Bibliotheken auf Basis von Linked Data Technologie*.[42] Der Kurzname AMSL soll nicht nur das Projekt, sondern auch das entwickelte ERM-System bezeichnen. Er ist kein Akronym, sondern spielt auf das bereits abgeschlossene Suchmaschinen-Projekt der UB Leipzig namens *finc* an.[43] Bei der Entwicklung von AMSL sollen die von der DLF ERMI ermittelten Anforderungen sowie die weiteren Aussagen des diese enthaltenden Abschlussberichtes berücksichtigt werden.[44] Als Projektziel wird „die Bereitstellung einer skalierbaren, nachnutzbaren Web-Applikation zur Verwaltung elektronischer Ressourcen in sächsischen Hochschulbibliotheken“[45] angegeben. Das geplante ERMS soll langfristig von den sächsischen Hochschulbibliotheken gemeinsam genutzt werden. Dazu ist eine zentrale Installation an der UB Leipzig vorgesehen, an deren einzelnen Wissensbasen den anderen Bibliotheken jeweilige Schreib- und Zugriffsrechte erteilt werden. In diesen privaten Bereichen können beispielsweise Daten zu Preisverhandlungen verwaltet werden, während am Jahresende zu teilende Informationen wie Berichte und Statistiken in einen für alle beteiligten Bibliotheken öffentlichen Bereich eingestellt werden können.[46]

Als Softwarebasis dient dazu das von AKSW entwickelte Applikationsframework OntoWiki, das in Kapitel *4.4.3 OntoWiki* ausführlicher erläutert wird. Zusammengefasst ist es damit möglich, eine Wissensbasis, die aus heterogenen Datenquellen zusammengesetzt ist, visuell darzustellen und auf

[38] Vgl. About AMSL [Elektronische Ressource].
[39] Vgl. Unterdörfel, Lydia: Entwicklung eines Electronic Resource Management Systems für Bibliotheken auf Basis von Linked Data Technologien [Elektronische Ressource], Folie 3.
[40] Vgl. Bitte testen: Alpha-Release unserer Datenmanagementsoftware [Elektronische Ressource].
[41] Vgl. Was ist d:swarm? [Elektronische Ressource].
[42] Vgl. About AMSL [Elektronische Ressource].
[43] Informationen aus persönlichem Gespräch mit Norman Radtke am 27.06.2014 in der UB Leipzig.
[44] Vgl. About AMSL [Elektronische Ressource].
[45] Ebd.
[46] Informationen aus persönlichem Gespräch mit Norman Radtke am 27.06.2014 in der UB Leipzig.

einfache Weise zu bearbeiten. OntoWiki besitzt zudem etliche Community-Features, was die geplante kooperative Arbeit unterstützt. Weitere Funktionalitäten, die für das ERMS benötigt werden, sollen durch im Rahmen des Projektes programmierte Erweiterungen ergänzt werden.[47]

Damit diese zunächst für die Hochschulbibliotheken in Sachsen entwickelte Lösung auch von anderen Bibliotheken adaptiert werden kann, wurde das Projekt und die Arbeit daran bereits auf nationalen wie auch internationalen Konferenzen wie dem 103. Bibliothekartag in Bremen und der ELAG (*European Library Automation Group*) 2014 in Bath (GB) vorgestellt[48].

Neben anderen Aufgaben besteht ein Teil der Projektarbeit darin, für verschiedene thematische Bereiche des ERMS bereits bestehende Vokabulare zu sichten und gegebenenfalls neu zu erarbeiten. Einzelne Teilgebiete betreffen beispielsweise administrative Informationen, Budgets und Rechnungen, Lizenzen und Verträge, Titel- und Produktinformationen, Nutzungsbedingungen sowie Nutzungsstatistiken.[49] Diese Aufgabe der Modellierung eines RDF-Vokabulars für Nutzungsstatistiken innerhalb des AMSL-Projektes wird durch die vorliegende Masterarbeit übernommen.

[47] Vgl. About AMSL [Elektronische Ressource].
[48] Vgl. News zu AMSL [Elektronische Ressource].
[49] Vgl. About AMSL [Elektronische Ressource].

3 Technischer Hintergrund

3.1 XML

Die *eXtensible Markup Language* XML ist, wie ihr Name bereits aussagt, eine Auszeichnungssprache. Sie dient der Darstellung hierarchisch strukturierter Daten, ist plattformunabhängig und, wie ebenfalls bereits am Namen erkennbar, erweiterbar. Eigene XML-Formate können unter anderem durch die Schemasprache XML-Schema definiert werden.[50] Entwickelt wurde XML ab 1996 aus dem bereits seit den 1980er Jahren bestehenden SGML (*Standard Generalized Markup Language*) und ist selbst seit 1998 eine *Empfehlung* (engl. *Recommendation*) des W3C.[51] Diese De-facto-Standards des W3C heißen zwar lediglich Empfehlung, haben aber trotzdem starken Einfluss auf die Entwicklung des Internets.

Da XML heutzutage eine der grundlegenden Sprachen nicht nur des Webs ist, wird im Folgenden eine gewisse Vertrautheit des Lesers mit XML vorausgesetzt und auf eine Erläuterung der zugrunde liegenden Konzepte und Details des Aufbaus verzichtet. An Stellen, an denen detailliert auf XML- und XML-Schema-Code eingegangen wird, werden die entsprechenden Elemente der Sprachen direkt knapp eingeführt.

Hier soll einzig kurz der Aspekt der Validierbarkeit von XML-Dokumenten in Erinnerung gerufen werden, da dieser in späteren Kapiteln dieser Arbeit im Vergleich zu Linked-Data-Vokabularen eine besondere Stellung einnimmt. Ein wohlgeformtes, das heißt mit den Syntaxregeln von XML konformes, Dokument wird noch nicht unbedingt als gültig bezeichnet. Dafür muss es außerdem auf eine ihm zugrunde liegende Grammatik wie etwa eine *Dokumenttyp-Definition* (DTD) oder ein XML-Schema verweisen und das darin definierte Format einhalten. Es ist also prinzipiell möglich, ein XML-Dokument auf gewisse Weise einzuschränken und anschließend die Einhaltung dieser Einschränkungen zu validieren.[52]

[50] Vgl. XML in 10 points [Elektronische Ressource].
[51] Vgl. ebd.
[52] Vgl. Wohlgeformtheit, Gültigkeit und Vollständigkeit einer XML-Datei [Elektronische Ressource].

3.2 Linked Data und Semantic Web

Wörtlich übersetzt bezeichnet der Begriff *Linked Data* zunächst nichts anderes als verknüpfte Daten. Die ebenfalls häufig anzutreffende und teils fälschlicherweise synonym gebrauchte Bezeichnung *Linked Open Data* ergänzt das Prinzip der Verknüpfung von Daten um deren Offenheit, also um die freie Verfügbarkeit dieser Daten unter einer offenen Lizenz.[53] Von dieser Verfügbarkeitskomponente abgesehen, bezeichnen Linked Data und Linked Open Data dasselbe Grundkonzept: Verknüpfte Daten. Der Anwendungskontext der verknüpften Daten führt zu einem weiteren, zunächst klärungsbedürftigen Begriff: dem *Semantic Web*.

Im Jahr 2001 veröffentlichten Tim Berners-Lee, Ora Lassila und Jim Hendler im *Scientific American* einen Artikel, der die Idee des Semantic Webs erstmals in breiterer Öffentlichkeit ins Gespräch brachte.[54] Das Semantic Web stellt eine Anwendung der Linked-Data-Technologien dar, anhand welcher die Grundkonzepte von Linked Data im Folgenden erläutert werden sollen. Die Autoren des *Scientific American*-Artikels schlugen vor, Daten im *World Wide Web* (WWW oder Web) nicht lediglich in einzelnen HTML-Dokumenten zu publizieren, sondern diese auch miteinander auf möglichst detaillierter Ebene zu verknüpfen. Berners-Lee nennt diese Erweiterung des Webs[55] den Schritt vom *Web of Documents* zum *Web of Data*.[56] In beiden Arten des Webs liegen Informationen in verteilten Dokumenten vor, doch im Web of Data sind nicht nur diese Dokumente, sondern die Informationen selbst durch *Uniform Resource Identifier* (URIs) einzeln identifizierbar.

Auch die Verknüpfungen erfolgen zwischen diesen einzelnen Informationen und mit Hilfe von URIs. Ein wesentlicher Unterschied der Verknüpfungen im Web of Data zu den konventionellen Hyperlinks im Web of Documents ist die Sinnbehaftung der Verbindungen. Statt lediglich anzugeben, *dass* zwei Informationen verknüpft sind, wird auch eine Aussage darüber getroffen, in welcher Beziehung diese zueinander stehen, also *wie* sie verknüpft sind. Die verwendeten URIs sollten möglichst dereferenzierbar sein, also mit Hilfe eines Internetprotokolls, üblicherweise HTTP, auf eine Repräsentation der identifizierten Daten verweisen. Dabei handelt es sich meist um HTML- oder XML-Dokumente, die die Ressource beschreiben und weitere Informationen sowie weitere Verweise auf verwandte Ressourcen liefern. Auf diese Weise ist es möglich, über Verweisungen und daran anschließende weitere Verweisungen große Netze sinnbehaftet verknüpfter und mit Metadaten angereicherter Daten zu erstellen.

[53] Vgl. Linked Data glossary [Elektronische Ressource]
[54] Vgl. Web 3.0 [Interview with Jim Hendler] [Elektronische Ressource].
[55] Vgl. W3C Semantic Web Frequently Asked Questions [Elektronische Ressource].
[56] Vgl. Linked Data [Elektronische Ressource].

Durch direkt miteinander verbundene Informationen über einzelne Entitäten und die Verwendung einheitlicher Standards wird das Web dabei auch für Maschinen verständlich: Wo das konventionelle Web lediglich einen Hyperlink zwischen zwei Dokumenten vorsieht, der von einem Menschen interpretiert werden muss, werden im Semantic Web zwei explizite Informationen durch eine beschriftete Verbindung verknüpft. Speziellen Software-Agents ist es möglich, diesen Verbindungen über mehrere Knotenpunkte hinweg zu folgen, dadurch verschiedene Informationen miteinander zu kombinieren und diese mit Hilfe ihrer Metadaten zu interpretieren.

Die Vorteile der Entwicklung für die verteilten Datenbestände des Semantic Webs kommen für Linked-Data-Technologien auch in anderen Anwendungsgebieten zum Tragen. Insbesondere wenn große Mengen unterschiedlich strukturierter, heterogener Daten aus verschiedenen Quellen zusammen verwaltet werden sollen, finden die feinen Verknüpfungsstrukturen für große, verteilte Datenmengen auch losgelöst vom Web Verwendung.[57] Um allerdings die einzelnen Konzepte der Linked-Data-Technologien in ihrem Aufbau zu verstehen, ist es stets notwendig, sich deren vorrangigen Einsatzort, das Semantic Web, und dessen Bedingungen vor Augen zu führen.

Einige Grundprinzipien im Aufbau des WWW wie auch des Semantic Webs prägen die Ansätze von Linked Data besonders stark. Das Web besteht prinzipiell aus verteilten Daten, die von beliebig vielen Benutzern erstellt werden. Im englischen Sprachraum wird diese Eigenschaft der Verteiltheit von Benutzern, Themen und Daten im sogenannte *AAA slogan* aufgegriffen: „Anyone can say Anything about Any topic."[58] *Anything* bedeutet in diesem Fall auch: Widersprüchliches. Es gibt im Web keine Instanz, die über Korrektheit von Daten entscheiden könnte. Aus dem AAA slogan folgt eine weitere Grundeigenschaft des Webs, nämlich die *Open World Assumption*. Dieses aus der formalen Logik stammende Konzept drückt vereinfacht gesagt aus, dass ein System von Aussagen niemals als vollständig betrachtet werden kann, sondern immer noch andere, wahre Aussagen existieren können.[59] Auch die Problematik des *Nonunique Naming* folgt aus dem AAA slogan. Da viele verschiedene Parteien an der Erstellung der Daten beteiligt sind, kann nicht davon ausgegangen werden, dass sie ihre Namensgebungsprozesse koordinieren.[60] Entsprechend ist es möglich, dass ein und dieselbe Information mehrfach unter unterschiedlichen Namen existiert. Diese Problematiken werden von den Linked-Data-Ansätzen zu berücksichtigen versucht, damit kein ungeordnetes „Datenchaos" entsteht.

[57] Vgl. W3C Semantic Web Frequently Asked Questions [Elektronische Ressource].
[58] Allemang, Dean: Semantic web for the working ontologist, S. 11.
[59] Vgl. ebd.
[60] Vgl. ebd., S. 12 f.

3.3 Grundkonzepte der Modellierung

Das Phänomen, dass sobald viele beliebige Parteien beliebige Aussagen zu beliebigen Themen treffen können, eine unübersichtliche Masse an sich eventuell widersprechenden Aussagen entsteht, ist weder neu noch auf das Semantic Web beschränkt. Es ist eine natürliche Entwicklung, wenn sich eine größere Gruppe von Menschen mit komplexen Themen beschäftigt. Die einfachste Lösung wäre es, eine übergeordnete Instanz eine präferierte Sichtweise auswählen zu lassen und diese anschließend für verbindlich zu erklären. Dieses Vorgehen ist im Web aufgrund dessen Konzipierung und generell pluralistischen Ansatzes sowie dem schlichten Fehlen einer solchen Instanz nicht zu verwirklichen. Eine weitere einfache Lösung wäre es, einzelne Sichtweisen parallel darzustellen und nicht miteinander zu verbinden. Dieser informelle Ansatz, der stets eines interpretierenden und einordnenden menschlichen Lesers bedarf, ist die derzeitige Realität im Web of Documents.[61] Für das die Maschinenlesbarkeit anstrebende Semantic Web kann ein solches Verfahren nicht zielführend sein.

Einen wirklichen Lösungsansatz können Modelle bieten. Denn diese sind in der Lage, gleiche Dinge zusammenzufassen und verschiedene voneinander zu trennen. Modelle bieten eine abstrakte Beschreibung eines Systems unter Auslassung gewisser Details und mit Betonung grundlegender Aspekte. Sie helfen, drei aufeinander aufbauende Problembereiche abzudecken: Sie unterstützen Menschen bei der Kommunikation, sie erklären Sachverhalte und lassen es dadurch zu, Vorhersagen zu treffen und sie vermitteln zwischen verschiedenen Betrachtungsweisen.[62]

Geht es in der zwischenmenschlichen Kommunikation um die Findung gemeinsamer Begrifflichkeiten, helfen bereits informelle Modelle, die beispielsweise die natürliche Sprache verwenden. Solche Modelle sind hoch flexibel und für Menschen leicht verständlich, aber durch die fehlenden Formalisierungen stets von der Interpretation des Nutzers abhängig.[63] Ein Formalismus lässt keinen Interpretationsspielraum mehr, sondern führt einen Sachverhalt in objektiver Form auf grundlegende Prinzipien und die diese verknüpfende Regeln zurück. Ein formales Modell bietet dadurch eine Erklärung des Sachverhaltes und kann mit Hilfe der zur Erklärung aufgestellten Regeln für die Vorhersage anderer Aussagen genutzt werden. . So sind beispielsweise in der Mathematik dieselben formalen Symboliken und Transformationen, die für den Beweis einer Aussage genutzt wurden, in der Lage, Ergebnisse andere Aussagen korrekt vorherzusagen.

[61] Vgl. Allemang, Dean: Semantic web for the working ontologist, S. 18.
[62] Vgl. ebd., S. 13.
[63] Vgl. ebd., S. 14 f.

Im Semantic Web wird ebenfalls mit Formalismen gearbeitet, sodass keine unterschiedlichen Interpretationsebenen entstehen und die Möglichkeit von Erklärungen und Vorhersagen gegeben ist. Das Bilden solcher Vorhersagen wird *Schließen* oder *Inferenzieren* (engl. *inferencing*) genannt. [64]

Doch auch ein bestimmtes formales Modell muss festgelegt werden. Da auch dazu jegliche übergeordnete Instanz fehlt, besagt die Grundidee, dass den gleichen Themenkomplex beschreibende Modelle aus verschiedenen Quellen als einzelne Ebenen eines großen Gesamtmodells aufgefasst werden.[65] Dadurch können alle verschiedenen Blickwinkel auf ein System zugleich modelliert und zueinander in Beziehung gesetzt werden.

Was Modelle außerdem voneinander unterscheidet, ist ihre Ausdrucksmächtigkeit (engl. *expressiveness* oder *expressivity*). Dabei ist ein tiefer gehendes Modell nicht zwingend das bessere, es handelt sich eher um „unterschiedliche Werkzeuge für verschiedene Zwecke.“[66] Auf dieser Ebene der Ausdrucksmächtigkeit lassen sich die drei Hauptsprachen bzw. Konzepte des Semantic Webs abgrenzen. RDF, das *Resource Description Framework*, dient dem Treffen von Aussagen und dem Zusammensetzen dieser zu einem einfachen Modell. RDFS (*RDF Schema*) erweitert dieses Datenmodell um die Ansätze von Gemeinsamkeit und Unterschiedlichkeit und hebt die Modellierung somit auf eine abstraktere Ebene. Die *Web Ontology Language* (OWL) schließlich bringt die Komponente der Logik mit in die Modellierung ein.[67]

3.4 RDF

3.4.1 Datenmodell

Das *Resource Description Framework* RDF wird seit 1997 durch das W3C als Kernkomponente des Semantic Webs entwickelt [68] und im Februar 1999 als grundlegende Empfehlung, der *RDF Model and Syntax Specification*, veröffentlicht.[69] Als aktuellste Version wurde am 25. Februar 2014 die Empfehlung *RDF 1.1 Concepts and Abstract Syntax* herausgegeben.[70].

[64] Vgl. Allemang, Dean: Semantic web for the working ontologist, S. 17 f.
[65] Vgl. ebd., S. 20.
[66] Ebd., S. 22.
[67] Vgl. ebd., S. 24.
[68] Vgl. Powers, Shelley: Practical RDF, S. 2.
[69] Vgl. Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation [Elektronische Ressource].
[70] Vgl. RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation [Elektronische Ressource].

Die Bedeutung von RDF für das Semantic Web vergleicht das W3C mit der Bedeutung von HTML für das herkömmliche Web, wobei RDF eine Struktur zur Beschreibung von Ressourcen und zur Einbindung von Metadaten darstellt.[71] Es ist ein Datenmodell, mit dessen Hilfe Aussagen über einzelne Entitäten getroffen und miteinander verbunden werden können. Aus diesen Verbindungen wiederum ist es möglich, bekannte wie auch neue Sinnzusammenhänge abzuleiten. Da RDF für das Web konzipiert ist, bestehen Namen in RDF aus URIs bzw. deren internationalisierter, im Zeichenvorrat erweiterter Variante, den *Internationalized Resource Identifiern* (IRI). Im Folgenden sollen die Grundprinzipien von RDF kurz erläutert werden.

In Anlehnung an die natürliche Sprache verwendet RDF Subjekt-Prädikat-Objekt-Konstruktionen. Eine Aussage wird also getroffen, indem einem Subjekt mit Hilfe eines Prädikates ein Objekt zugeordnet wird. Auf diese Art formulierte Daten werden als *Tripel* (engl. *triple*) bezeichnet und in Tripelstores genannten, speziell für dieses Format optimierten Datenbanken verwaltet. Neben der Interpretation als Tripel bietet sich aber auch die graphische Darstellung der Daten als beschriftete, gerichtete Graphen (engl. *Labeled Directed Graphs*) an.[72] Dabei werden die als Ovale (bzw. als Rechtecke, dazu unten mehr) dargestellten Knotenpunkte (engl. *nodes*) durch beschriftete Pfeile (engl. *arcs*) miteinander verbunden.[73] Im unteren Beispiel sagt der RDF-Graph aus, dass das Subjekt „Tolkien“ das Objekt „DerHerrDerRinge“ „schreibt“, also der Autor des Werkes ist.

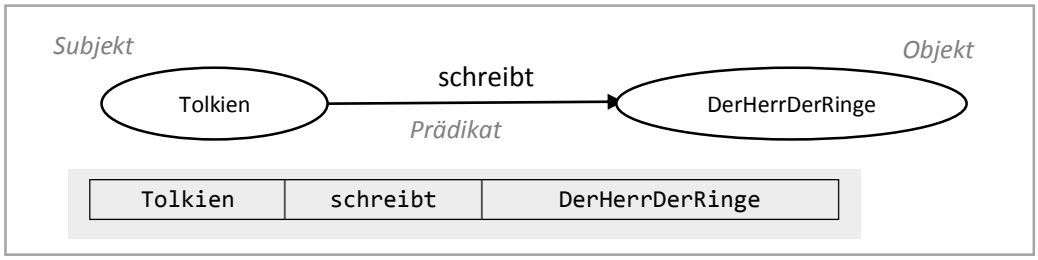


***Abb. 2** Einfacher gerichteter und bezeichneter RDF-Graph.*

[71] Zitat des W3C nach Powers, Shelley: Practical RDF, S. 1.
[72] Vgl. Das Graphenmodell von RDF [Elektronische Ressource], S. 2.
[73] Vgl. Powers, Shelley: Practical RDF, S. 19.

Jedes RDF-Tripel aus Subjekt, Prädikat und Objekt stellt für sich einen einzigartigen, vollständigen Fakt dar. Es kann mit andern Tripeln verbunden werden, indem das Objekt des einen Tripels zum Subjekt eines weiteren Tripels wird.[74] Beispielsweise wurde der Herr der Ringe nicht nur von Tolkien geschrieben, sondern spielt auch in der fiktiven Welt Mittelerde. Der untere RDF-Graph fügt entsprechend die folgende Aussage hinzu: „DerHerrDerRinge“, nun das Subjekt, „spieltIn“ „Mittelerde“.

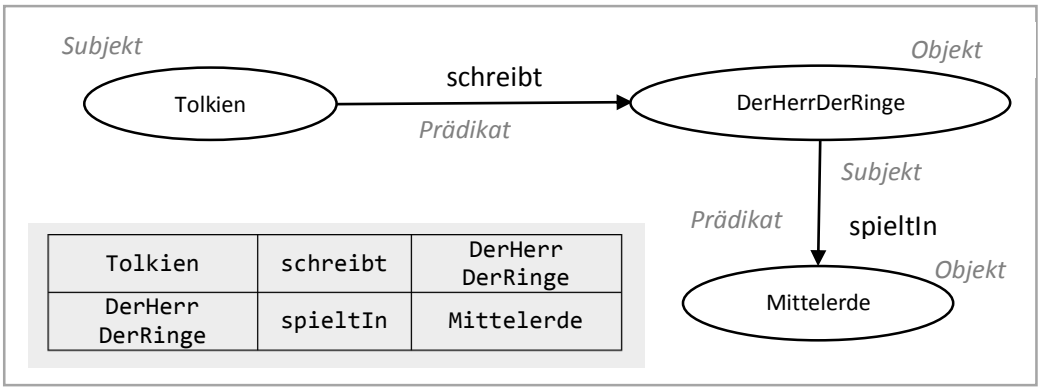


Abb. 3 *RDF-Graph mit zwei Pfeilen. Objekt eines Tripels wird zum Subjekt eines zweiten Tripels.*

Dabei ist zu beachten, dass jedes der verbundenen Tripel für sich immer ein eigener, vollständiger Fakt mit seiner eigenen Bedeutung bleibt, unabhängig von den Aussagen des Gesamtmodells.[75] Dies ist den Grundkonzepten des Semantic Webs geschuldet: Nach der Open World Assumption ist es stets möglich, dass mehr Informationen zu einem Thema existieren, als in einem Modell enthalten sind. Dieses Modell muss trotzdem, auch ohne Kenntnis der weiteren Aussagen, interpretierbar sein.

Bei der Art der Daten, die Subjekt, Prädikat und Objekt bilden können, unterscheidet RDF die Ressourcen in *Individuen* (engl. *Individuals*) mit URI-Referenzen und *Literale* (engl. *Literals*). Individuen verfügen stets über URI-Referenzen und sind dadurch, wie bereits der Name sagt, immer über URIs identifizierbar und damit sowohl eindeutig voneinander unterscheidbar als auch näher beschreibbar. Literale dagegen sind lediglich Zeichenketten, die nicht näher bestimmt und auch nicht zur weiteren Verknüpfung genutzt werden können. Entsprechend müssen Subjekt und Prädikat stets Individuen sein. Nur Objekte können entweder Individuen oder Literale sein. Zur Unterscheidung in der graphischen Darstellung werden Individuen als Ovale gezeichnet, Literale dagegen als Rechtecke.[76]

[74] Vgl. Powers, Shelley: Practical RDF, S. 18.
[75] Vgl. ebd., S. 18 f.
[76] Vgl. ebd., S. 19 ff..

In RDF ist es schließlich auch möglich, Ressourcen einzuführen, die über keinen URI verfügen und auch kein Literal sind. Solche Ressourcen sind beispielsweise nötig, wenn Aussagen über Dinge getroffen werden sollen, deren Identität nicht bekannt ist oder die lediglich eine Art von Gegenstand darstellen, keinen Gegenstand selbst. In dem Beispiel des Autors Tolkien könnte eine Aussage zu treffen sein, dass Tolkien Autor von Fantasy-Büchern ist. Da aber die einzelnen Fantasybücher Tolkiens für diese Aussage nicht näher spezifiziert werden können, wird eine unbenannte Ressource, ein sogenannter *Blanknode* eingeführt, die im Graphen durch ein leeres Oval repräsentiert wird.[77]

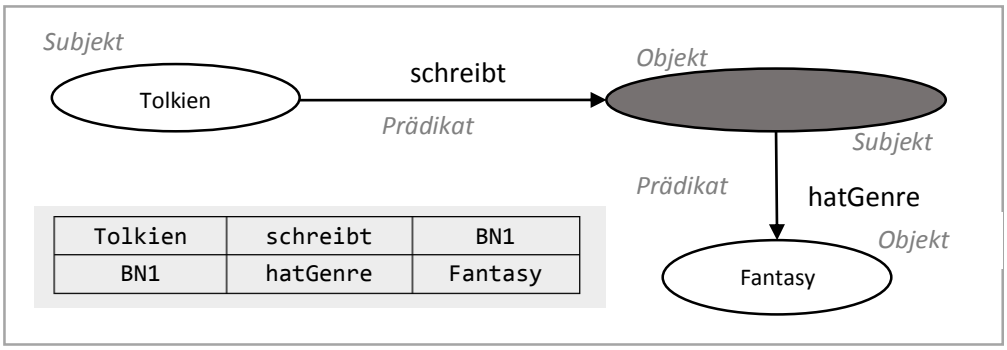


***Abb. 4** RDF-Graph mit Blanknode.*

Für die Benennung von Ressourcen existieren einige Regelungen des W3C, die als De-facto-Standards angesehen werden. Allgemein sind demnach Ressourcen in der sogenannten *CamelCase*-Schreibweise anzusetzen, [78] um durch Leerzeichen entstehenden Kompatibilitätsproblemen vorzubeugen und ein einheitliches Erscheinungsbild zu begünstigen.[79] CamelCase bedeutet, eine aus mehreren Wörtern bestehende Bezeichnung zusammenhängend in Kleinbuchstaben zu schreiben und jedes neu beginnende Wort durch einen großen ersten Buchstaben, also durch Binnenversalien, zu kennzeichnen.[80] Die so entstehenden Ausdrücke erinnern an die Höcker eines Kamels, worin sich der englische Name der Schreibweise begründet.[81] CamelCase wird, je nach der Schreibung des ersten Buchstabens, unterschieden in *UpperCamelCase* und *lowerCamelCase*. Für die Benennung von Klassen[82] und Individuen soll möglichst UpperCamelCase verwendet werden, für Eigenschaften lowerCamelCase.[83] Neben der Großschreibung ist für die Benennung von Ressourcen auch der Numerus zu beachten: Unabhängig vom Kontext soll generell die Singularform genutzt werden.[84]

[77] Vgl. Powers, Shelley: Practical RDF, S. 20.
[78] Vgl. Allemang, Dean: Semantic web for the working ontologist, S. 310.
[79] Vgl. CamelCase [Elektronische Ressource].
[80] Vgl. Allemang, Dean: Semantic web for the working ontologist, S. 310.
[81] Vgl. CamelCase [Elektronische Ressource].
[82] Zu Klassen und Eigenschaften mehr in Kapitel *3.5 RDFS*.
[83] Vgl. Allemang, Dean: Semantic web for the working ontologist, S. 311.
[84] Vgl. ebd.

3.4.2 Serialisierungen

RDF selbst ist lediglich ein Datenmodell und keine Auszeichnungssprache, die bisher genutzten Darstellungsformen als Graphen oder Tabellen sind für eine Publikation im Web ungeeignet. Um RDF-strukturierte Daten ins Netz zu bringen, sind folglich entsprechende textuelle Sprachen nötig, die als RDF-Serialisierungssprachen bezeichnet werden.[85] Es gibt etliche, teils aufeinander aufbauende Serialisierungssprachen für RDF, von denen in dieser Masterarbeit aufgrund ihrer Einfachheit lediglich die schreib- und lesefreundliche *Terse RDF Triple Language* (Turtle) zum Einsatz kommt und entsprechend nachfolgend kurz erläutert werden soll. Um den Grad der Schreibvereinfachungen von Turtle zu illustrieren, folgt darauf ein kurzes Beispiel der für die Kompatibilität mit Web-Infrastrukturen konzipierten[86] XML-Serialisierung *RDF/XML*.

Turtle-Dateien haben die Erweiterung `.ttl`, RDF/XML-Dateien die Erweiterung `.rdf`. Es ist mittels Parser/Serializer-Tools, wie etwa der UNIX-basierten *Raptor RDF Syntax Library* (Raptor)[87], nicht nur möglich, RDF-Tripel aus serialisierten Dateien zu extrahieren, sondern auch, in Turtle erstellten Code zunächst in Tripel und dann auf diesen basierend in RDF/XML umzuwandeln und umgekehrt.

Turtle ist inzwischen eine Empfehlung des W3C und wird für die manuelle Erstellung von RDF-Dateien empfohlen.[88] Es basiert auf dem Grundgedanken, RDF-Tripel in der Reihenfolge Subjekt - Prädikat - Objekt hintereinander und nur durch Leerzeichen getrennt in eine Zeile zu schreiben. Da diese Praxis aber aufgrund der Länge der URIs sehr unübersichtlich ist, bedient sich Turtle der ursprünglich für XML eingeführten, sogenannten *Qualified Names* (Qnames). Dabei handelt es sich um ein Abkürzungsprinzip für URIs, das einen URI in einen *Namensraum* (engl. *namespace*) und einen innerhalb dieses Namensraums eindeutigen *Identifikator* (engl. *Identifier*) abkürzt.[89] Statt des langen URIs `http://www.example.com/example#tolkien` könnte also der wesentlich kürzere QName `ex:tolkien` genutzt werden. Dabei repräsentiert `ex:` den Namespace und `tolkien` stellt den Identifikator dar. Zu Beginn einer Turtle-Datei muss selbstverständlich zunächst der Namensraum deklariert und dabei in den vollständigen URI übersetzt werden, damit der Identifikator seine Eindeutigkeit auch im globalen Web beibehält.[90] Nach der folgenden Turtle-Deklaration ist eine Verwendung von QNames für die obigen RDF-Beispiele möglich

```
@prefix ex: <http://www.example.com/example#>
```

[85] Vgl. Allemang, Dean: Semantic web for the working ontologist, S.44.
[86] Vgl. ebd., S. 46.
[87] Erreichbar unter: http://librdf.org/raptor/, mehr dazu in Kapitel *4.4.2 Raptor*.
[88] Vgl. RDF 1.1 Turtle, W3C Recommendation [Elektronische Ressource].
[89] Vgl. Allemang, Dean: Semantic web for the working ontologist, S. 36.
[90] Vgl. ebd., S. 45.

Das erste Tolkien-Beispiel schreibt sich in Turtle danach wie folgt:

```
ex:Tolkien ex:schreibt ex:DerHerrDerRinge .
```

Subjekt, Prädikat und Objekt stehen, durch ihre QNames abgekürzt, in einer Zeile, die mit einem Punkt abgeschlossen wird. Literale, die aufgrund ihres fehlenden URIs keinen QName benötigen, werden in Anführungszeichen geschrieben.[91]

```
ex:DerHerrDerRinge ex:veröffentlicht "1954" .
```

Turtle bietet außer den QNames etliche weitere, sehr schreib- und lesefreundliche Verkürzungen.[92] Da alle in dieser Arbeit verwendeten RDF-Beispiele von nun an in Turtle verfasst sind, werden nachfolgend einige der wichtigsten Verkürzungen vorgestellt.

- Besitzen mehrere aufeinanderfolgende Tripel das gleiche Subjekt, kann dieses ab dem zweiten Tripel weggelassen werden. Dafür müssen alle zusammenhängenden Zeilen, die das gleiche Subjekt haben, mit einem Semikolon statt des Punktes enden. Am Schluss dieses Tripelblockes steht ein Punkt:[93]

```
ex:Tolkien ex:schreibt ex:DerHerrDerRinge ;
       ex:geborenIn ex:Suedafrika ;
       ex:gestorbenIn ex:England .
```

- Unterscheiden sich mehrere Tripel lediglich im Objekt, so können die verschiedenen Objekte durch Kommata getrennt werden:[94]

```
ex:Tolkien ex:schreibt ex:Hobbit, ex:Silmarillion .
```

- Ein Blanknode wird ausgedrückt, indem alle Tripel, deren Subjekt er ist, in eckige Klammern geschrieben werden. Dieser eingeklammerte Ausdruck kann daraufhin als Objekt eines anderen Tripels genutzt werden.[95] Dass Tolkien in Mittelerde spielende Fantasy-Literatur verfasst, kann also wie folgt ausgedrückt werden:

```
ex:Tolkien ex:schreibt [ ex:hatGenre ex:Fantasy ;
                         ex:spieltIn ex:Mittelerde ] .
```

[91] Vgl. RDF 1.1 Turtle, W3C Recommendation [Elektronische Ressource].
[92] Einsehbar unter http://www.w3.org/TR/2014/REC-turtle-20140225/.
[93] Vgl. RDF 1.1 Turtle, W3C Recommendation [Elektronische Ressource].
[94] Vgl. ebd.
[95] Vgl. ebd.

- Die Klassenzuweisung[96] `rdf:type` einer Ressource kann, analog zur englischen Normalsprache, durch den Artikel `a` als Verkürzung von „*is a*“ geschrieben werden, woraufhin sich der Ausdruck eher wie ein natürlicher Satz lesen lässt:[97]

```
ex:Tolkien a ex:Autor .
```

Für die Kompatibilität mit Web-Infrastrukturen empfiehlt das W3C die Nutzung der XML-Serialisierung von RDF, RDF/XML.[98] Diese ist für den menschlichen Bearbeiter zwar unübersichtlicher als Turtle, lässt sich aber besser in die XML-basierte Landschaft des WWW einbinden. Auf die genaue Syntax von RDF/XML[99] soll hier aus Platzgründen nicht eingegangen werden. Folgendes Beispiel des Tripels

```
Tolkien    schreibt    Hobbit
```

soll genügen, um einen Eindruck des prinzipiellen Aufbaus zu gewinnen:

```
<rdf:RDF xmlns:ex="http://www.example.com/example#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about="http://www.example.com/example#
     Tolkien">
        <ex:schreibt rdf:resource="http://www.example.com/example#
          Hobbit"/>
    </rdf:Description>
</rdf:RDF>
```

3.5 RDFS

RDF Schema ist seit 2004 eine Empfehlung des W3C, die aktuellste Version ist die Empfehlung *RDF 1.1 Schema* vom 25. Februar 2014, die RDFS an das neue RDF 1.1 Dokumentenset anpasst, das unter anderem auch RDF 1.1 und RDF 1.1 Turtle enthält.[100] Die Sprache RDFS ist selbst in RDF verfasst und definiert RDF-Ressourcen, die zur Beschreibung anderer RDF-Ressourcen verwendet werden können.[101] Ein gültiges RDFS-Dokument ist dadurch immer auch ein gültiges RDF-Dokument,[102] wodurch schematische Zuordnungen mit Hilfe von RDFS-Ressourcen innerhalb des eigentlichen

[96] Mehr zu Klassen in Kapitel *3.5 RDFS*.
[97] Vgl. RDF 1.1 Turtle, W3C Recommendation [Elektronische Ressource].
[98] Vgl. Allemang, Dean: Semantic web for the working ontologist, S. 46
[99] Einsehbar unter http://www.w3.org/TR/rdf-syntax-grammar/.
[100] Vgl. RDF 1.1 Schema, W3C Recommendation [Elektronische Ressource].
[101] Vgl. ebd.
[102] Vgl. Allemang, Dean: Semantic web for the working ontologist, S. 127.

RDF-Dokuments gemacht werden können. Dazu muss lediglich der Namespace, in dem die RDFS-Ressourcen durch den Standard definiert werden, verknüpft werden. Er wird in QNames im Allgemeinen durch das Präfix `rdfs:` abgekürzt und durch den folgenden URI identifiziert:

```
http://www.w3.org/2000/01/rdf-schema#
```

Inhaltlich ist RDFS eine semantische Erweiterung von RDF, die Mechanismen zur Beschreibung von Gruppen verwandter Ressourcen und deren Beziehungen zueinander einführt.[103] RDFS-Dokumente beschreiben also Sachverhalte auf der abstrakteren Ebene der Beziehungen zwischen Klassen und Eigenschaften. Dieses Class/Property-System basiert darauf, dass Eigenschaften durch die Angabe der Klassen jener Ressourcen beschrieben werden, auf welche sich die Eigenschaften beziehen.

Da in einem Tripel immer das Prädikat eine Eigenschaft ist, sind die zu beschreibenden Ressourcen also das Subjekt und das Objekt des Tripels. Diese Beschreibung geschieht über die Ressourcen `rdfs:domain` und `rdfs:range`. Wenn ausgesagt wird, dass eine Property P die Domain D bzw. die Range R hat, bedeutet dies, dass wann immer die Property P in einem Tripel als Prädikat verwendet wird, geschlossen werden kann, dass das Subjekt des Tripels Element der Klasse D bzw. das Objekt des Tripels Element der Klasse R ist.[104] Das Subjekt bzw. das Objekt des Tripels wird also durch diesen Domain/Range-Mechanismus einer Klasse zugeordnet.

Ein nicht zu vernachlässigender Punkt dabei ist, dass es sich nicht um eine gewöhnliche Einschränkung handelt, die im Falle einer Inkonsistenz zu einem Fehler führen würde. RDFS kennt keine ungültigen Eingaben, wie es bei anderen Schemasprachen wie etwa XML Schema der Fall ist, denn es wird generell keine Aussage über die Gültigkeit der verbundenen Daten gemacht. Statt durch Domain und Range festzulegen, dass nur gewisse Klassen durch die entsprechende Eigenschaft verknüpft werden könnten, lässt RDFS eine Verknüpfung beliebiger Klassen zu und schlussfolgert die notwendigen Informationen, um weiterhin ein konsistentes Modell darzustellen.[105]

Diese Erläuterung der Ressourcen `rdfs:domain` und `rdfs:range` soll genügen, um die prinzipielle Struktur von RDFS zu verdeutlichen. Weitere RDFS-Ressourcen wie etwa `rdfs:subClassOf` werden im Modellierungsteil dann eingeführt, wenn sie gebraucht werden.

[103] Vgl. RDF 1.1 Schema, W3C Recommendation [Elektronische Ressource].
[104] Vgl. Allemang, Dean: Semantic web for the working ontologist, S. 130.
[105] Vgl. ebd., S. 131.

Für die graphische Darstellung von RDFS werden, da gültiges RDFS immer auch gültiges RDF ist, die Konventionen für die Darstellung von RDF-Graphen beibehalten. Demnach werden die Knotenpunkte des Graphen durch Ovale und Rechtecke dargestellt, Ovale für Individuen, Rechtecke für Literale. Die Eigenschaften werden von beschrifteten Pfeilen repräsentiert.[106] Aus Gründen der Übersichtlichkeit werden in dieser Arbeit Ressourcen, die Eigenschaften darstellen, hellgrau eingefärbt und Ressourcen, die Klassen darstellen, dunkelgrau.

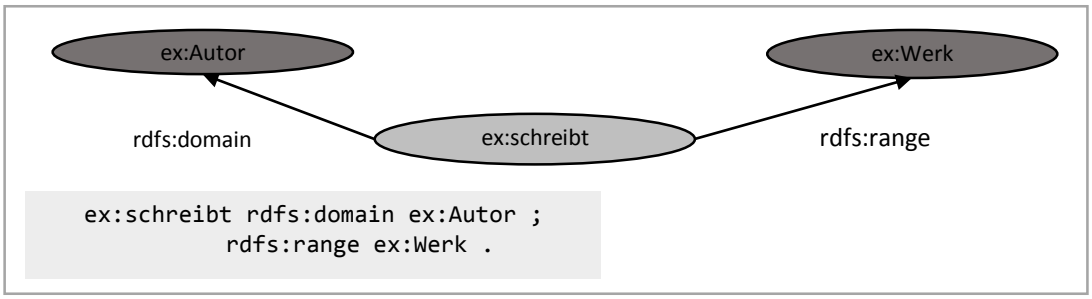


Abb. 5 *Darstellung einer Domain/Range-Beziehung durch zwei Tripel.*

3.6 OWL

Die *Web Ontology Language*, als Akronym nicht in der richtigen Reihenfolge „WOL" abgekürzt, sondern „OWL" wie Eule, ist seit 2004 eine Empfehlung des W3C.[107] Die aktuelle Version ist die OWL 2-Spezifikation aus dem Jahr 2012,[108] eine Überarbeitung des 2009 eingeführten OWL 2.[109] Wie auch bei RDF und RDFS bestehen die Namen in OWL aus URIs bzw. IRIs. OWL kann durch verschiedene Syntaxen ausgedrückt werden, unter anderem auch durch die RDF-basierten Serialisierungen RDF/XML und Turtle.[110] Letztere wird in dieser Arbeit zum Einsatz kommen.

Der große Mehrwert von OWL im Vergleich zu RDFS ist die Möglichkeit, Logik auszudrücken. OWL ist keine Schemasprache mehr, sondern eine Ontologiesprache.[111] Das Konzept der *Ontologie* (engl. *ontology*) ist in verschiedensten wissenschaftlichen Disziplinen beheimatet, seien es Philosophie,

[106] Vgl. Powers, Shelley: Practical RDF, S. 20 f.
[107] Vgl. Allemang, Dean: Semantic web for the working ontologist, S. 24.
[108] Vgl. OWL Web Ontology Language Current Status [Elektronische Ressource].
[109] Vgl. Allemang, Dean: Semantic web for the working ontologist, S. 24.
[110] Vgl. OWL 2 Web Ontology Language Primer (Second Edition), W3C Recommendation [Elektronische Ressource].
[111] Vgl. Allemang, Dean: Semantic web for the working ontologist, S. 24.

Linguistik oder eben Informatik. Allen diesen Formen von Ontologien gemeinsam ist die exakte formale Erfassung von menschlichem Wissen.[112] Dabei werden präzise, beschreibende Aussagen über die Welt oder Teile dieser Welt formuliert. Der entsprechende Teil wird als das Gebiet der Ontologie bzw. deren Domäne bezeichnet.[113] Neben diesen auf spezielle Gebiete ausgerichteten Ontologien existieren auch solche, die ohne ein konkretes Anwendungsgebiet versuchen, möglichst umfassend die Bedeutung allgemeiner Begrifflichkeiten zu erklären.[114] In der Philosophie und anderen Geisteswissenschaften dient die formale Beschreibung von Ontologien hauptsächlich dazu, ein Verständnis für die Kategorisierung von Objekten zu gewinnen. Die eigentliche Ontologie ist eher ein Nebenprodukt als das Ziel der Arbeit. Dies ist ein entscheidender Unterschied zu Ontologien in der Informatik, wo die erstellte Ontologie selbst, als Produkt der Formalisierung, das Ziel ist. Aus dieser tatsächlichen Anwendung der Ontologien erwächst die Notwendigkeit von Technologien, um diese zu formulieren und zu verwalten. [115] Dazu trägt unter anderem OWL als Ontologiesprache bei.

Trotz der wesentlich größeren Ausdrucksmächtigkeit sind die OWL zugrundeliegenden Denkweisen weitgehend deckungsgleich mit den bereits erläuterten Ansätzen von RDFS. OWL ist eine deklarative Sprache, die logische Aussagen trifft, um einen bestimmten Wissensstand zu beschreiben. Mit Hilfe sogenannter *Reasoner* kann später aus den getroffenen Aussagen neues Wissen gefolgert werden.[116] Obwohl einige Teile von OWL zunächst diesen Anschein erwecken, ist auch OWL nicht in der Lage, Forderungen an den syntaktischen Aufbau eines Dokumentes zu stellen oder direkte Beschränkungen für Daten zu treffen. Dies und die stets präsente Open World Assumption unterscheiden OWL, genau wie Linked Data allgemein, stark von konventionellen Datenbanksystemen. Bei der Modellierung mit OWL muss immer davon ausgegangen werden, dass nicht bekanntes Wissen trotzdem vorhanden und möglicherweise wahr ist.[117]

OWL bietet, je nach gewünschter Expressivität, drei aufeinander aufbauende Sprachebenen, die jeweils gewisse Eigenarten besitzen und insgesamt unterschiedlich starke Einschränkungen der Funktionalitäten von OWL darstellen. OWL Lite ist am stärksten eingeschränkt, OWL DL (kurz für *description logic*) hat ebenfalls noch Einschränkungen und OWL Full bietet den gesamten Funktionsvorrat und die volle Ausdrucksmächtigkeit. Die Einschränkungen sollen das Entwickeln von Tools wie etwa Reasonern vereinfachen.[118] Es muss nicht zu Anfang eines OWL-Dokuments angegeben

[112] Vgl. Stuckenschmidt, Heiner: Ontologien, S. V.
[113] Vgl. OWL 2 Web Ontology Language Primer (Second Edition), W3C Recommendation [Elektronische Ressource].
[114] Vgl. Stuckenschmidt, Heiner: Ontologien, S. 3.
[115] Vgl. ebd., S. 93.
[116] Vgl. OWL 2 Web Ontology Language Primer (Second Edition), W3C Recommendation [Elektronische Ressource].
[117] Vgl. ebd.
[118] Vgl. ebd.

werden, welche Variante genutzt wird, dies ergibt sich aus der Verwendung von Funktionen der entsprechenden Varianten.

Während im vorangegangenen Kapitel grundlegende Konzepte von RDFS erläutert wurden, wird an dieser Stelle darauf verzichtet, da die Prinzipien von OWL wesentlich komplexer sind als jene von RDF und RDFS. Einzelne OWL-Konstrukte werden stattdessen direkt im Modellierungsteil am Beispiel ihrer konkreten Anwendung eingeführt und erläutert.

TEIL II - Modellierung

4 Vorarbeiten

4.1 Anforderungsanalyse

Es soll eine Möglichkeit gefunden werden, im XML-Format abgerufene COUNTER-Reports in das im Linked-Data-Format organisierte ERMS der UB Leipzig einzubinden. Die COUNTER-Reports enthalten Informationen, die bisher noch mittels keinem einheitlichen Vokabular darstellbar sind. Deshalb ist eine sofortige Konvertierung der Daten nach RDF nicht möglich ist, sondern es muss ein den Bedürfnissen angepasstes Vokabular entwickelt werden. Entsprechend wurden in mehreren Gesprächen mit Mitarbeitern des AMSL-Projektes die Vorstellungen von einem idealen Vokabular diskutiert, was die Basis der nachfolgenden Überlegungen darstellt.

Einige Eigenschaften und Klassen, insbesondere solche, die die im COUNTER-Report ebenfalls enthaltenen organisatorischen und bibliographischen Daten modellieren, könnten auch aus anderen Vokabularen nachgenutzt werden, wie es ein für Linked Data übliches Verfahren ist. Es besteht allerdings der Wunsch der UB Leipzig, zunächst ein gänzlich eigenständiges Vokabular mit exakt auf die Bedürfnisse der COUNTER-Report-Verwaltung abgestimmten Klassen und Eigenschaften zu entwerfen, das eine exakte Wiedergabe aller in den XML-Reports enthaltenen Daten ermöglicht. Zu einem späteren Zeitpunkt können in dieses Vokabular auch im Nachhinein noch passende Ressourcen anderer Vokabulare eingebettet bzw. die eigenen Klassen auf diese bezogen werden.

Die genaueren Anforderungen an das zu erstellende, eigenständige Vokabular können in funktionale bzw. Verhaltensanforderungen, nicht-funktionale bzw. Qualitätsanforderungen sowie in Problembereichsanforderungen unterteilt werden.[119] Im Sinne der besseren Nachvollziehbarkeit[120] wird bei jeder Anforderung angegeben, aus welchen Gründen sie gestellt wird.

[119] Vgl. Knauber, Peter: Anforderungsanalyse [Elektronische Ressource], Folie 27.
[120] Vgl. ebd., Folie 41.

Verhaltensanforderungen beschreiben Dienste, die das System zur Verfügung stellen soll sowie Reaktionen des Systems auf bestimmte Eingaben.[121] Direkte Funktionen im Sinne einer automatisierten Reaktion auf Nutzereingaben kann ein RDF-Vokabular grundsätzlich nicht haben, da es kein ausführbares Programm darstellt. Entsprechend kann im Folgenden nur in Anlehnung an das aus der Software-Entwicklung stammende Schema vorgegangen werden. Dem Prinzip von Verhaltensanforderungen am ehesten zuzuordnen sind die folgenden, die direkte Funktionalität des Vokabulars betreffenden Punkte:

1.1 Vollständige Abbildung der XML-COUNTER-Reports ohne Informationsverlust.

Die Kernfunktion an das zu erstellende Vokabular ist die Abbildung der XML-COUNTER-Reports. Diese Abbildung soll vollständig und ohne Informationsverlust erfolgen, das heißt, dass alle in der XML-Datei enthaltenen, den Report betreffenden Informationen übernommen werden sollen. Diese Forderung folgt aus dem Ziel, langfristig ein Alternativformat zu XML anzubieten. Für die konkrete Nutzung des Vokabulars innerhalb des ERMS wäre eine Übernahme aller Informationen aus den Reports nicht unbedingt notwendig.

1.2 Sinnvolle Navigation zwischen Reportbestandteilen.

Eine weitere, als funktional zu bezeichnende Kernanforderung an das Vokabular ist es, sinnvolle Verbindungen zwischen den einzelnen Bestandteilen eines Reports herzustellen. Dadurch soll in der Anwendung eine nutzerfreundliche Navigation durch die Nutzungsdaten sowie eine effiziente Datenabfrage ermöglicht werden.

1.3 Ständige Zuordnung Daten/Report möglich.

Eine weitere Anforderung ist es, dass die Zuordnung der Daten zu ihren Reports erhalten bleibt. Dies soll zur Transparenz der Herkunft der Daten beitragen und eventuelle Nachfragen an den Datenanbieter vereinfachen.

Die nächste Anforderungskategorie sind die Qualitätsanforderungen. Diese beschreiben Elemente des Systems, die keine direkten Funktionen sind.[122]

2.1 Deutsche und englische Erläuterungen.

Die Anforderung, dass das RDF-Vokabular mit deutschen und englischen Erläuterungen zu versehen ist, fällt in diese Kategorie. Ziel dieser Forderung ist eine möglichst geringe Einschränkung des Nutzungskreises einerseits, was die Übernahme der englischen Sprache des COUNTER-Schemas bedingt, sowie die komfortable

[121] Vgl. Knauber, Peter: Anforderungsanalyse [Elektronische Ressource], Folie 27.
[122] Vgl. ebd.

Nutzung durch Mitarbeiter der UB Leipzig andererseits, weshalb auch deutsche Bezeichnungen eingeführt werden.

Die letzte Kategorie umfasst die Problembereichsanforderungen. Diese entstammen der Domäne der späteren Anwendung des Systems,[123] im Falle des Vokabulars also der bibliothekarischen Nutzung.

3.1 Linked Data auf Basis von RDF.

Die Kernanforderung, die aus der Anwendung erwächst, ist das bereits zu Anfang erwähnte und geforderte Linked-Data-Format. Da die Nutzungsstatistiken in das gesamte ERMS integriert werden sollen, das auf Linked-Data-Technologie basiert, ist dieses Format auch für die COUNTER-Reports verpflichtend.

3.2 Wiedergabe möglichst vieler Informationen aus dem XML-Schema.

Wie es bei der Nutzung von Linked Data häufig der Fall und einfach durchführbar ist, soll das Vokabular auf einfache Weise auch für andere Nutzer verfügbar gemacht werden, die eventuell andere Einsatzzwecke verfolgen. Diese anderen, bisher unbekannten Aufgaben sollen insofern berücksichtigt werden, als dass das Vokabular nicht zu eng an die Verwendung im ERMS der UB Leipzig angepasst werden soll. Unter dieser Betrachtungsweise ergibt es außerdem Sinn, nach Möglichkeit auch die im XML-Schema vorhandenen Einschränkungen umzusetzen. Dadurch kann das Vokabular beispielsweise einfacher auch zur nativen RDF-Erstellung von COUNTER-Reports genutzt werden und nicht nur zur Konvertierung bereits XML-konformer Reports.

3.3 Berücksichtigung von Besonderheiten von OntoWiki.

Da die UB Leipzig für die Realisierung des ERMS den Einsatz der Software OntoWiki plant, sind deren eventuelle Besonderheiten und sich daraus ergebende Konsequenzen für das Vokabular nach Möglichkeit zu berücksichtigen.

3.4 Ressource als Ausgangspunkt.

Aus der bibliothekarischen Nutzungspraxis ergibt sich weiterhin die Anforderung, Statistiken über die Nutzung einer Ressource möglichst direkt von dieser Ressource ausgehend abrufen zu können, da ein Großteil der Fragestellungen darauf beruht, Daten zu einer bestimmten Ressource einzusehen.

[123] Vgl. Knauber, Peter: Anforderungsanalyse [Elektronische Ressource], Folie 27.

Bei der Konzeption des RDF-Vokabulars sollen diese Anforderungen möglichst vollständig umgesetzt und mit detaillierten, konkretisierten Inhalten gefüllt werden. Wie es allgemein von Anforderungen an ein System erwartet wird,[124] besteht für alle genannten Anforderungen prinzipiell die Möglichkeit, nach der Modellierung eine erfolgreiche Umsetzung mit Hilfe von Testdaten zu überprüfen.

4.2 Analyse des COUNTER XML-Schemas

4.2.1 Grundstruktur

Das in der Datei *counter4_0.xsd* definierte COUNTER XML-Schema ist darauf angelegt, langfristig für alle verschiedenen COUNTER-Reports verwendbar zu sein. Entsprechend ist es in seiner Konzeption eher allgemein gehalten.[125] Um bei Aktualisierungen des Code of Practise nicht dieses von den Anwendern genutzte allgemeine XML-Schema überarbeiten zu müssen, [126] werden die für einige Elemente explizit festgelegten, zugelassenen Werte nicht innerhalb der eigentlichen Schemadatei selbst definiert. Stattdessen sind zum einen die für die Bezeichnung von Reports zu verwendenden Werte in der *SUSHI Reports Registry*[127] aufgelistet, einer auf der Website der NISO zugänglichen Tabelle. Die restlichen Elemente mit vorgegebenen Werten werden in der separaten XML-Schema-Datei *counterElements4_0.xsd* definiert. Über das Element `xs:include` wird diese zweite Schemadatei in die erste eingebunden und dem gemeinsamen Target-Namespace auf den Servern der NISO zugeordnet.[128] Als Präfix für diesen Namespace wird, als Abkürzung für COUNTER, der Buchstabe c definiert:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:c="http://www.niso.org/schemas/counter"
targetNamespace="http://www.niso.org/schemas/counter">
    <xs:include schemaLocation="counterElements4_0.xsd"/>
    …
</xs:schema>
```

[124] Vgl. Knauber, Peter: Anforderungsanalyse [Elektronische Ressource], Folie 45.
[125] Vgl. Schemas for the Standardized Usage Statistics Harvesting Initiative (SUSHI) [Elektronische Ressource].
[126] Vgl. COUNTER Schema Data Element Values [Elektronische Ressource].
[127] Abrufbar unter http://www.niso.org/workrooms/sushi/reports.
[128] Vgl. XML Schema include Element [Elektronische Ressource].

In seiner Struktur orientiert sich das COUNTER XML-Schema an der Erstellung der Reports durch den Datenanbieter. Die Informationen beziehen sich auf drei unterschiedliche Gebiete. Es werden neben den eigentlichen Statistiken sowohl organisatorische, d. h. auf die beteiligten Akteure bezogene, als auch die konkrete Publikation betreffende Daten erfasst. Diese drei Gebiete werden allerdings im XML-Schema nicht explizit voneinander getrennt betrachtet, sondern gehen ineinander über. Zunächst soll der grobe hierarchische Aufbau des XML-Schemas aufgezeigt werden, ohne auf alle vorhandenen Typen und Elemente einzugehen.

Ausgehend vom Wurzelelement `c:Reports` (Plural) kann eine XML-Datei mehrere einzelne COUNTER-Reports enthalten, die, wie etwa *Journal Report*, *Book Report* oder *Database Report*, ihrerseits Daten über entsprechende Publikations- und Zugriffsarten enthalten. Für jeden dieser einzelnen Reports steht ein Element `c:Report` (Singular), das ein Kindelement von `c:Reports` ist. Der jeweilige einzelne Report gliedert sich weiter auf in die Elemente `c:Vendor` und `c:Customer`, für den Datenanbieter und den Kunden. Dabei werden unter dem Datenanbieter tatsächlich nur organisatorische Informationen wie Namen und Kontaktadressen verzeichnet. Unter `c:Customer` dagegen werden neben diesen organisatorischen auch alle weiteren Daten zu Ressource und Nutzung zusammengefasst. Es ist dabei möglich dass ein Report mehrere Kunden mit ihren jeweiligen Daten auflistet, was für Consortium Reports von Bedeutung ist.

Dem Element c:`Customer` sind also alle weiteren Daten hierarchisch untergeordnet. Als Kindelement folgt zunächst das beliebig oft wiederholbare Element `c:ReportItems`, welches, obwohl es im Plural steht, immer nur eine einzelne Ressource bezeichnet, also etwa eine Zeitschrift oder ein

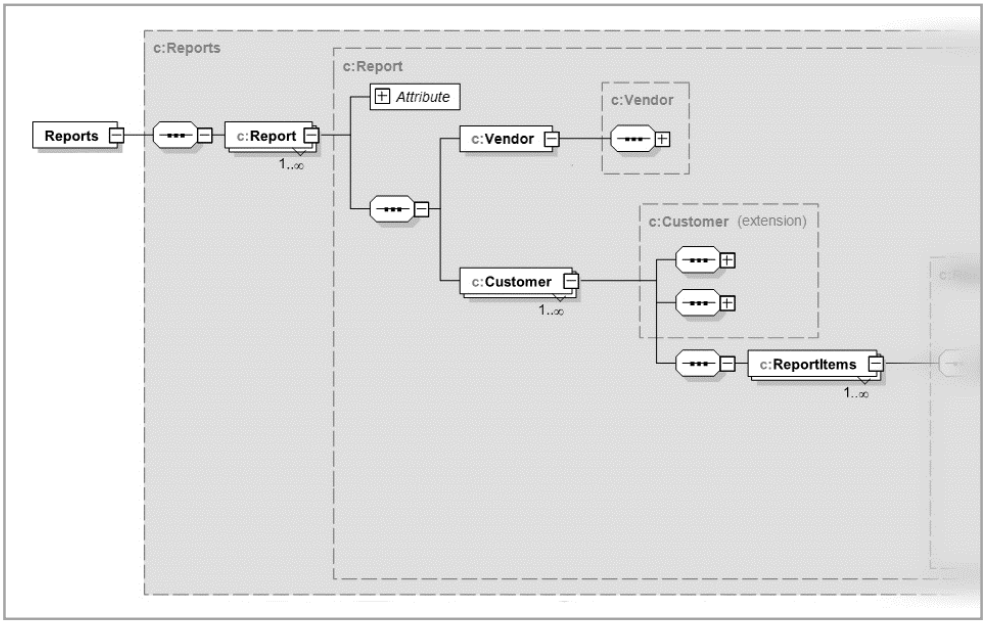


Abb. 6
XML-Schema, Überblick, Teil 1.

E-Book, über dessen Nutzung berichtet wird. Den einzelnen Medien unter `c:ReportItems` werden zunächst die ressourcenspezifischen Metadaten wie Identifikationsnummer, Medienart, Titel und Verlag zugeordnet. Die statistischen Nutzungsdaten werden über ein weiteres, beliebig oft wiederholbares Kindelement namens `c:ItemPerformance` mit der Ressource verknüpft.

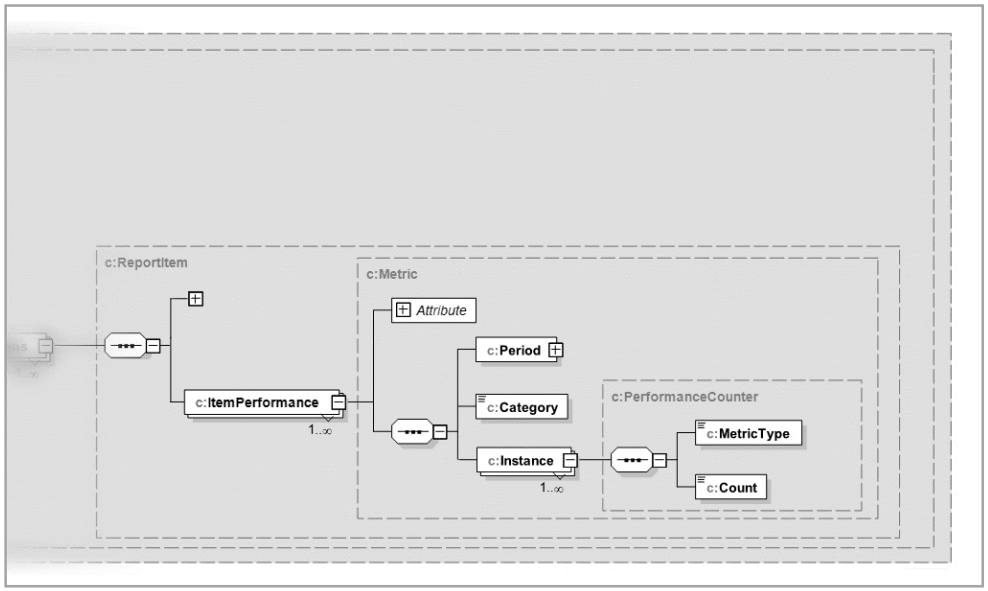


Abb. 7
XML-Schema, Überblick, Teil 2.

Unter einem solchen `c:ItemPerformance`-Element wird jeweils ein Messzeitraum von einem Monat zusammengefasst. Außerdem wird jeweils nur eine Messkategorie verzeichnet, also etwa nur die Anfragen an die Datenbank oder nur die verweigerten Zugriffe. Der Messzeitraum wird über das Element `c:Period` erfasst, die Messkategorie über das Element `c:Category`. Beide Elemente dürfen pro `c:ItemPerformance`-Element nur einmal auftreten.

Das dritte Kindelement ist dagegen wiederholbar. Es trägt den Namen `c:Instance` und besitzt zwei Kindelemente. Das erste, `c:MetricType`, umfasst die gemessene Größe, wie etwa PDF-Volltext oder Abstract. Dabei ist es nicht wiederholbar: für jede Messgröße muss ein neues `c:Instance`-Element erstellt werden. Das zweite Kindelement trägt den Namen `c:Count` und umfasst schließlich den eigentlichen Zählstand als ganze Zahl.

Eine typische, vom Anbieter abgerufene COUNTER-XML-Datei enthält also mehrere Reports. Diesen ist jeweils der Anbieter als `c:Vendor` zugeordnet. Zu einem Report gehören meist ein, eventuell aber auch mehrere Lizenznehmer als `c:Customer`. Jedem einzelnen Lizenznehmer sind dessen jeweils lizenzierte Medien zugeordnet. Für jedes einzelne Medium findet sich für jeden Monat und für jede Kategorie ein `c:ItemPerformance`-Element. Werden also über ein Jahr hinweg sowohl die Anfragen als auch die verweigerten Zugriffe gemessen, existieren zu jeder Ressource jeweils 24 `ItemPerformance`-Elemente. Wird nun wiederum immer in PDF- und HTML-Volltexte sowie ge-

samte Volltexte unterschieden, besitzt jedes der 24 `c:ItemPerformance`-Elemente drei `c:Instance`-Elemente, die jeweils den eigentlichen Zahlenwert enthalten. In diesem Beispiel wären also zu jeder lizenzierten Ressource insgesamt 72 Zahlenwerte verzeichnet. Bei einigen Hundert lizenzierten Ressourcen und mehreren Reports pro abgerufener XML-Datei ergeben sich auf diese Weise umfangreiche Dateien.

4.2.2 Details

In der Beschreibung der Grundstruktur wurde bisher nur auf den groben inhaltlichen Aufbau, aber weder auf die Details noch auf die konkrete Syntax des XML-Schemas eingegangen. Um die Übertragung in das RDF-Vokabular nachvollziehbar zu halten, ist eine entsprechend detaillierte Beschreibung zumindest von exemplarischen Teilen des XML-Schemas allerdings notwendig.

Wie bereits aus dem grundlegenden Aufbau ersichtlich, enthalten viele der Elemente in den COUNTER-XML-Dateien weitere Elemente. Einfache XML-Elemente können allerdings keine anderen Elemente enthalten, dafür sind komplexe Elemente erforderlich. Entsprechend existieren *komplexe Datentypen* (engl. *complex types*), die andere Elemente und Attribute enthalten können[129] und in einer Schemadatei definiert werden. Das COUNTER-XML-Schema definiert insgesamt 13 komplexe Typen. Davon werden im Folgenden exemplarisch der komplexe Typ `c:Metric` sowie die in dessen Definition gebrauchten weiteren komplexen Typen, Elemente und Attribute ausführlich erläutert. Damit ist der statistische Teil des Schemas komplett abgedeckt.

Auf das restliche Schema wird an dieser Stelle nicht genauer eingegangen, da die wichtigsten Syntax-Elemente bereits durch das gewählte Beispiel erläutert werden. Die gesamte Schemadatei *counter4_0.xsd* ist im elektronischen Anhang D sowohl in Codeform als auch in graphischer Darstellung enthalten, die ergänzende Schemadatei *counterElements4_0.xsd* ist in Codeform angehängt.

Der vorzustellende komplexe Typ `c:Metric` umfasst die statistischen Daten aus dem COUNTER-Report. Er ist im Schema unter Angabe seines Namens definiert:

```
<xs:complexType name="Metric"> … </xs:complexType>
```

[129] Vgl. XML Schema complexType Element [Elektronische Ressource].

Um Elemente zu kommentieren und dabei die Nachvollziehbarkeit der Schemadatei zu verbessern, wird das Element `xs:documentation` verwendet, welches immer innerhalb eines `xs:annotation`-Elements auftreten muss.[130] Das Parent-Element `xs:annotation` kann in jedem anderen Element auftreten,[131] so etwa in `xs:complexType`, wo es in dem Falle des komplexen Typs `c:Metric` aussagt, dass dieser verwendet wird, um die eigentlichen Nutzungsdaten zu beschreiben:

```
<xs:complexType name="Metric">
    <xs:annotation>
        <xs:documentation>
            The actual usage details.
        </xs:documentation>
    </xs:annotation>
…
</xs:complexType>
```

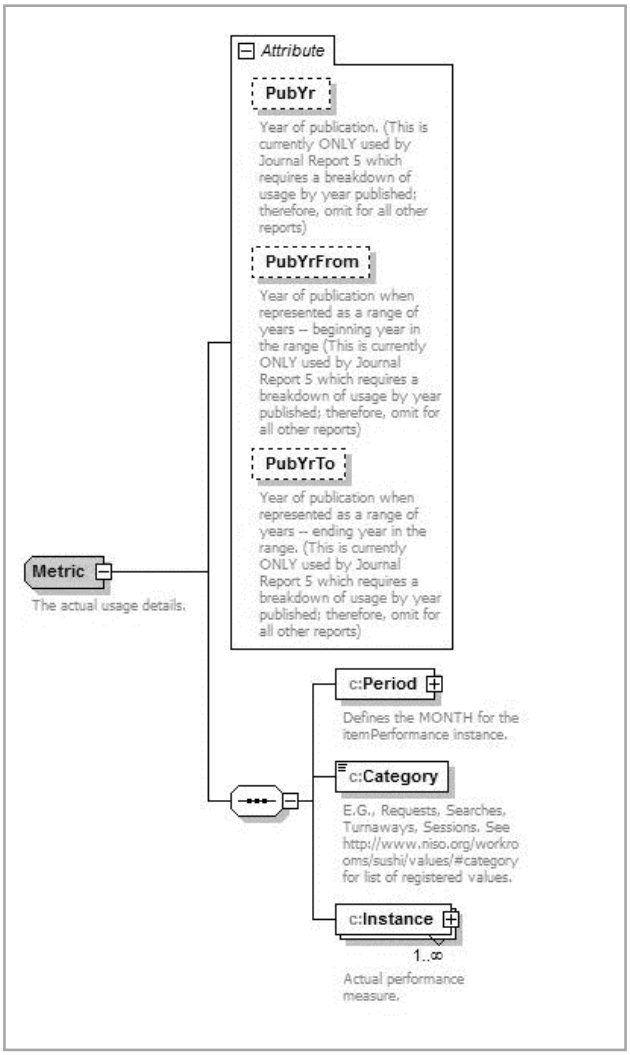


Abb. 8
XML-Schema, Grafik, komplexer Typ `c:Metric`.

[130] Vgl. XML Schema documentation Element [Elektronische Ressource].
[131] Vgl. XML Schema annotation Element [Elektronische Ressource].

Außer der lediglich erläuternden Annotation umfasst `c:Metric` die drei in einer Sequenz angeordneten Kindelemente `c:Period`, `c:Category` und `c:Instance`, die weiter unten beschrieben werden sollen, sowie die drei Attribute `c:PubYr`, `c:PubYrFrom` und `c:PubYrTo`. Inhaltlich dienen diese Attribute dazu, die in *Journal Report 5* jeweils einzeln betrachteten Archivjahrgänge von Zeitschriften zu unterscheiden,[132] indem unter dem Attribut `c:PubYr` das Publikationsjahr der Ressource angegeben wird. Werden die Nutzungsdaten für mehrere Publikationsjahre zusammengefasst wiedergegeben, werden statt `c:PubYr` die beiden anderen Attribute genutzt, von denen `c:PubYrFrom` das Start- und `c:PubYrTo` das Endjahr der anzugebenden Zeitspanne angeben.

Attribute selbst sind generell *einfache Datentypen* (engl. *simple types*), können aber nur bei komplexen Elementen auftreten, nicht bei einfachen Elementen.[133] Definiert wird ein Attribut unter Angabe von mindestens seinem Namen, hier "PubYr", und seinem Datentyp, hier `xs:gYear`.[134] Außerdem werden alle drei Attribute von `c:Metric` durch jeweils ein `xs:annotation/documentation`-Element beschrieben.

```
<xs:attribute name="PubYr" type="xs:gYear">
    <xs:annotation>
        <xs:documentation>
            Year of publication.
        </xs:documentation>
    </xs:annotation>
</xs:attribute>
```

Attribute sind standardmäßig nur optional zu benutzen. Soll die Nutzung eines Attributes gefordert werden, so muss in der Attributdefinition das Attribut `xs:use` verwendet werden:[135]

```
<xs:attribute name="PubYr" type="xs:gYear" use="required"/>
```

Da die Attribute von `c:Metric` inhaltlich aber nur für *Journal Report 5* relevant sind und nur dort verwendet werden,[136] sind sie in dem für alle Reports konzipierten, übergreifenden Schema daher nur optional zu nutzen, das `xs:use`-Attribut entfällt.

Die drei Kindelemente von `c:Metric` sind, wie oben erwähnt, in einem `xs:sequence`-Element angeordnet. Dies bedeutet, dass sie immer in der vorgegebenen Reihenfolge auftreten müssen. Für die Anzahl des Auftretens gibt es keine grundsätzliche Begrenzung. Jedes der Kindelemente kann

[132] Vgl. The COUNTER Code of Practice for e-Resources: Release 4 [Elektronische Ressource], S. 12.
[133] Vgl. XSD Attributes [Elektronische Ressource].
[134] Vgl. ebd.
[135] Vgl. XSD Attributes [Elektronische Ressource].
[136] Vgl. The COUNTER Code of Practice for e-Resources: Release 4 [Elektronische Ressource], S. 12.

gar nicht oder beliebig häufig auftreten, wenn dies über die Attribute xs:minOccur und xs:maxOccur für minimale bzw. maximale Häufigkeit entsprechend angegeben wird. Die bei Nichtverwendung dieser Attribute geltenden Standardwerte sind jeweils eins, ein solches Kindelement muss also genau einmal verwendet werden.[137]

Das erste Element in der Sequenz trägt den Namen c:Period und ist vom komplexen Datentyp c:DateRange, der im Schema definiert und weiter unten in diesem Kapitel näher erläutert wird. Es muss genau einmal auftreten und definiert den Berichtsmonat.

```
<xs:sequence>
    <xs:element name="Period" type="c:DateRange">
        <xs:annotation>
            <xs:documentation>
                Defines the MONTH for the ItemPerformance
                instance.
            </xs:documentation>
        </xs:annotation>
    </xs:element>
    …
</xs:sequence>
```

Darauf folgt das Element c:Category, das die Art der gesammelten Nutzungsdaten angibt. Es ist vom Datentyp c:Category, einem einfachen Datentyp, welcher nicht in der eigentlichen Schema-Datei definiert ist, sondern in der zu Anfang erwähnten, verknüpften Datei *counterElements4_0.xsd*. Der einfache Typ c:Category wird durch eine Einschränkung des Datentyps xs:string definiert. Dafür wird ein xs:restriction-Element genutzt, das über das Attribut xs:base den Basisdatentyp als xs:string definiert. Dann folgt durch eine Aufzählung mit Hilfe von xs:enumeration-Elementen die eigentliche Einschränkung von allen möglichen Werten vom Typ xs:string auf nur die in der Aufzählung enthaltenen. [138]

```
<xs:simpleType name="Category">
      …
    <xs:restriction base="xsd:string">
        <xs:enumeration value="Requests"/>
        <xs:enumeration value="Searches"/>
        <xs:enumeration value="Access_denied"/>
    </xs:restriction>
</xs:simpleType>
```

[137] Vgl. XML Schema sequence Element [Elektronische Ressource].
[138] Vgl. XML Schema restriction Element [Elektronische Ressource].

Dieser abschließenden Aufzählung im COUNTER Elements-Schema entsprechend sind als Werte "Requests", "Searches" und "Access_denied" zugelassen. Laut den das Schema ergänzenden Tabellen auf den Websites der NISO sind die genannten Werte jeweils nur bei bestimmten Arten von Ressourcen bzw. den diesen zugeordneten c:ItemDataTypes anzuwenden. Diese sind in Tabelle 1 aufgeführt. [139]

Category-Wert	Kompatibler ItemDataType
Requests	Journal, Book, Multimedia, Collection
Searches	Database, Platform, (Service)[1]
Access denied	Journal, Book, Database, (Service)

Tab. 1
Einfacher Typ Category und zugelassene Werte

Das dritte Element der Sequenz trägt den Namen c:Instance und ist vom komplexen Typ c:PerformanceCounter, der unten näher erläutert wird. Durch die Festlegung des Attributs xs:maxOccurs darf es mehr als einmal auftreten:

```
<xs:sequence>
      …
    <xs:element name="Instance" type="c:PerformanceCounter"
    maxOccurs="unbounded">
        …
    </xs:element>
    …
</xs:sequence>
```

Damit sind alle Elemente des komplexen Datentyps c:Metric definiert. Wie oben erwähnt, sind aber einige der verwendeten Elemente selbst von einem komplexen Datentyp, der ebenfalls im Schema definiert wird. Hierbei handelt es sich um die Typen c:DateRange und c:PerformanceCounter.

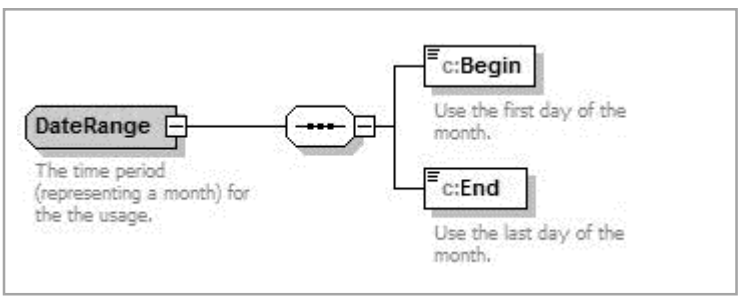


Abb. 9
XML-Schema, Grafik, komplexer Typ DateRange.

[139] Vgl. COUNTER Schema Data Element Values [Elektronische Ressource].

Der Typ c:DateRange gibt die Zeitspanne an, über die hinweg die berichtete Nutzung gemessen wurde. Diese Zeitspanne muss immer einen Monat umfassen, dessen erster und letzter Tag angegeben wird. Dazu werden in einer Sequenz zwei Kindelemente definiert. Beide sind vom Datentyp xs:date und sollen genau einmal genutzt werden. Das erste Element trägt den Namen c:Begin und gibt den ersten Tag des Berichtsmonats an, das zweite Element heißt c:End und gibt entsprechend den letzten Tag des Berichtsmonats an.

Der andere verwendete komplexe Typ, c:PerformanceCounter, wird im Schema als „unused" kommentiert, obwohl er wie oben gesehen als Datentyp des Elements c:Instance genutzt wird. Diese offenbar fehlerhafte Angabe wird im Folgenden entsprechend übergangen.

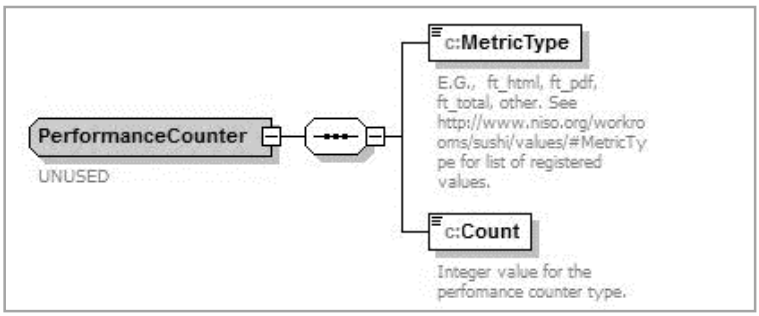


Abb. 10
XML-Schema, Grafik, komplexer Typ `PerformanceCounter`.

Der Typ c:PerformanceCounter umfasst die konkreten statistischen Zahlen. In einer Sequenz werden ihm zwei Kindelemente zugeordnet, die jeweils genau einmal zu verwenden sind. Das erste Kindelement, c:MetricType, ist vom gleichnamigen Datentyp c:MetricType, der analog zu dem oben erwähnten Typ c:Category ebenfalls ein in der Datei *counterElements4_0.xsd* durch abschließende Aufzählung definierter, auf xs:string basierender einfacher Datentyp ist. Er enthält die gemessene Größe bzw. das gemessene Format. Die genauen Bedeutungen der zugelassenen Werte sind auf den Seiten der NISO erläutert[140] und in Tabelle 2 ausschnittsweise zusammengefasst. Das zweite Element in der Sequenz hat den Namen c:Count und ist vom Typ xs:nonNegativeInteger. Dieses Element enthält schließlich den Zahlenwert der entsprechenden Nutzungsmessung.

Die anderen komplexen und einfachen Datentypen, Elemente und Attribute, die im COUNTER-Schema definiert werden, sind weitestgehend mit Hilfe ähnlicher Strukturen beschrieben. Sie können den in Anhang D aufgeführten Dateien und graphischen Darstellungen entnommen werden.

[140] Vgl. COUNTER Schema Data Element Values [Elektronische Ressource].

MetricType-Wert	Kompatibler ItemDataType
ft_ps	Volltextanfragen von Postscript-Dateien
ft_ps_mobile	Volltextanfragen von Postscript-Dateien, optimiert für mobile Endgeräte
ft_pdf	Volltextanfragen von PDF-Dateien
ft_pdf_mobile	Volltextanfragen von PDF-Dateien, optimiert für mobile Endgeräte
ft_html	Volltextanfragen von HTML-Versionen
ft_html_mobile	Volltextanfragen von HTML-Versionen, optimiert für mobile Endgeräte
ft_epub	Volltextanfragen von epub-Formaten
sectioned_html	Genutzt in Book Report 2, Consortium Report 1 and Title Report 1 zur Unterscheidung des Ausmaßes (extent) des gelieferten Buchinhaltes
ft_total	Summe aller Volltextanfragen
toc	Anfragen von Inhaltsverzeichnissen
abstract	Anfragen von Artikel-Abstracts bzw. anderer detaillierter Metadaten ohne Volltext
reference	Anfragen von bibliographischen Metadaten zu einem Artikel
…	…

Tab. 2 *Ausschnitt der für den einfachen Typ* `MetricType` *zugelassenen Werte.*

4.3 Grundkonzeption

Das Kernproblem der Konzeption ist eine sinnvolle Übertragung der im XML-Schema enthaltenen Elemente in ein RDF-Vokabular. Hierbei sollen die Vorteile von RDF möglichst umfassend genutzt werden und gleichzeitig der Einsatzzweck des Vokabulars innerhalb des ERM-Systems der UB Leipzig berücksichtigt werden. Um auch für zukünftige Anwendungen keine Informationen zu verlieren, sollen entsprechend der Anforderungsanalyse stets alle in einer XML-Datei enthaltenen Informationen in die zu erstellende RDF-Datei übernommen werden.

Während das XML-Schema von den abgerufenen Reports ausgehend Informationen in immer kleinere, hierarchisch einander untergeordnete Einheiten bündelt, ist die Stärke von RDF, einzelne Elemente miteinander sinntragend zu verknüpfen. Eine strenge Hierarchie ist hier nicht nötig und auch eine Übernahme der durch das XML-Schema vorgegebenen Logik ist nicht zwingend.
Eine wichtige festgestellte Problembereichsanforderung ist die Anpassung an die Arbeitsweisen und Fragestellungen in der bibliothekarischen Umgebung. Die abgerufenen Reports dienen in der Praxis dazu, Nutzungsdaten für bestimmte Ressourcen zu erhalten. In der bibliothekarischen Arbeit stellen sich insbesondere Fragen zu der Nutzung bestimmter Ressourcen, Plattformen oder Dateiformaten in gewissen Zeiträumen. *Wie viele Nutzer hatte die Zeitschrift x im Februar 2014? Werden Zeitschriften des Abonnements y häufiger im PDF- oder im HTML-Format gelesen? Wird das E-*

Book-Paket z immer noch genauso wenig genutzt wie im Vorjahr? Der eigentliche Report und seine ursprüngliche Strukturierung sind eher sekundär. Wichtig ist, dass eine eindeutige Zuordnung von Nutzungsdaten und Ressource sowie den entsprechenden Zusatzinformationen gegeben ist und die vorhandenen Informationen aus möglichst vielen Blickwinkeln auffindbar gemacht werden können.

Folglich soll im zu formulierenden RDF-Vokabular nicht mehr der Report in das Zentrum der Betrachtungen gestellt werden, sondern das Reportelement, das die ausgewertete Ressource repräsentiert. Dieses Reportelement stellt den Anknüpfungspunkt zu den übrigen Funktionen des ERMS dar, wie etwa den bibliographischen Daten oder der Lizenzverwaltung.

Die im XML-Schema nur angedeutete Teilung in die Akteure und die Ressource betreffende sowie konkrete statistische Daten wird im RDF-Vokabular deutlicher. Der zentrale Komplex betrifft die Ressource. In dessen Mittelpunkt steht die Klasse `:ReportItem`, die hier, wie die anderen wichtigen Klassen auch ‚zunächst nur erwähnt und später detailliert vorgestellt werden soll. Direkt auf die jeweilige Ressource bezogen ist der Themenkomplex der statistischen Daten mit der Klasse `:CountingInstance` im Mittelpunkt. Ein weiter inhaltlicher Komplex besteht aus den organisatorischen, akteurbezogenen Daten, bei denen die beiden Klassen `:Vendor` und `:Customer` die wichtigsten sind. Einen vierten inhaltlichen Komplex bilden im RDF-Vokabular die auf den Report bezogenen Daten. Zwischen den einzelnen inhaltlichen Komplexen existieren Verbindungen, um die jeweiligen Daten miteinander in Beziehung zu setzen.

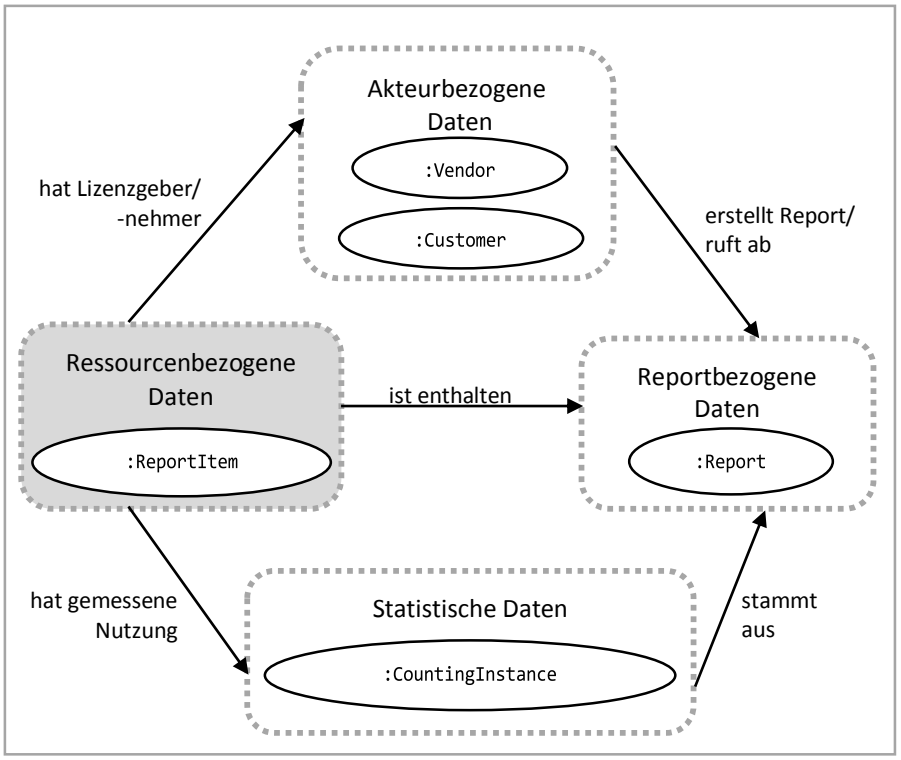


Abb. 11
Prinzipieller Aufbau des RDF-Vokabulars, inhaltliche Teilgebiete, wichtige Klassen.

Ein bisher nicht erwähnter, aber überaus wichtiger Unterschied zwischen XML-Report und REDF-Repräsentation ist, dass die RDF-Daten langfristig im ERMS verbleiben. Während ein XML-Report stets für sich allein steht, müssen die RDF-Daten im Kontext bereits vorhandener Informationen aus anderen Reports betrachtet werden. Dadurch ergibt sich die Schwierigkeit, dass einige Informationen, insbesondere die ressourcen- und akteurbezogenen Daten, häufig über mehrere Reports hinweg identisch bleiben, sich aber im Laufe der Zeit auch, wie etwa ein überarbeiteter Titel einer Zeitschrift, ändern können. Diese Veränderungen müssen in Form einer Art Historie abbildbar gemacht werden, da im Sinne der geforderten Transparenz nicht lediglich der bisherige Name überschrieben werden darf, sondern zu jedem beliebigen Zeitpunkt der entsprechende Name feststellbar sein muss. Eine Lösung des Problems ist das Treffen einer Aussage über eine Aussage, das hier angewandt werden soll.

4.4 Verwendete Programme

4.4.1 Notepad++

Da Turtle-Code durch seine Verkürzungen nicht nur lese- sondern auch schreibfreundlich ist und aus reinem Text-Code besteht, ist ein spezieller Editor prinzipiell nicht erforderlich. Dennoch ist eine syntaxspezifische Farbkennzeichnung des Überblicks wegen nützlich. Diese wird von dem Editor *Notepad++* als optionale Erweiterung angeboten.

Notepad++ ist ein freier Texteditor, der unter anderem eine Syntax-Hervorhebung für etliche Programmier- und Auszeichnungssprachen anbietet. Der in C++ geschriebene[141] Quellcode steht unter der GPL-Lizenz und wurde hauptsächlich von Don Ho erstellt.[142] Durch verschiedene kleine Features[143] eignet sich der auch mit deutschem Interface verfügbare Editor sehr gut für die Erstellung von Turtle-Code.

Abb. 12 *Notepad++ Logo.*

[141] Vgl. Notepad++ [Elektronische Ressource].
[142] Vgl. Notepad++ Contributors [Elektronische Ressource].
[143] Vgl. Notepad++ Features [Elektronische Ressource].

Besonders hilfreich ist die angebotene Syntax-Hervorhebung, die Schlüsselwörter der erkannten bzw. eingestellten Sprache farblich hervorhebt und so zur Übersichtlichkeit beiträgt. Turtle ist zwar keine standardmäßig erkennbare Sprache, wird aber als *User Defined Language File* zur nachträglichen Erweiterung zum Download bereitgestellt.[144] Durch die standardmäßige XML-Unterstützung ist Notepad++ auch für die oberflächliche Bearbeitung von XML-Dateien nützlich, wenn auch vom Funktionsumfang nicht mit spezialisierten, kommerziellen Anwendungen wie *XMLSpy* vergleichbar.

Notepad++ bietet weiterhin eine Auto-Vervollständigung an, die bei der Eingabe schon vorhandene Ausdrücke anbietet. Dies ist insbesondere bei wiederholt auftretenden QNames zeitsparend. Ebenfalls praktisch, allerdings weniger im Umgang mit Turtle- als mit XML-Dateien, ist die Syntax-Faltung. Dieses Feature erlaubt, logisch zusammenhängende Inhalte durch Einklappen zu minimieren und trägt so zur Übersichtlichkeit des Dokumentes bei.

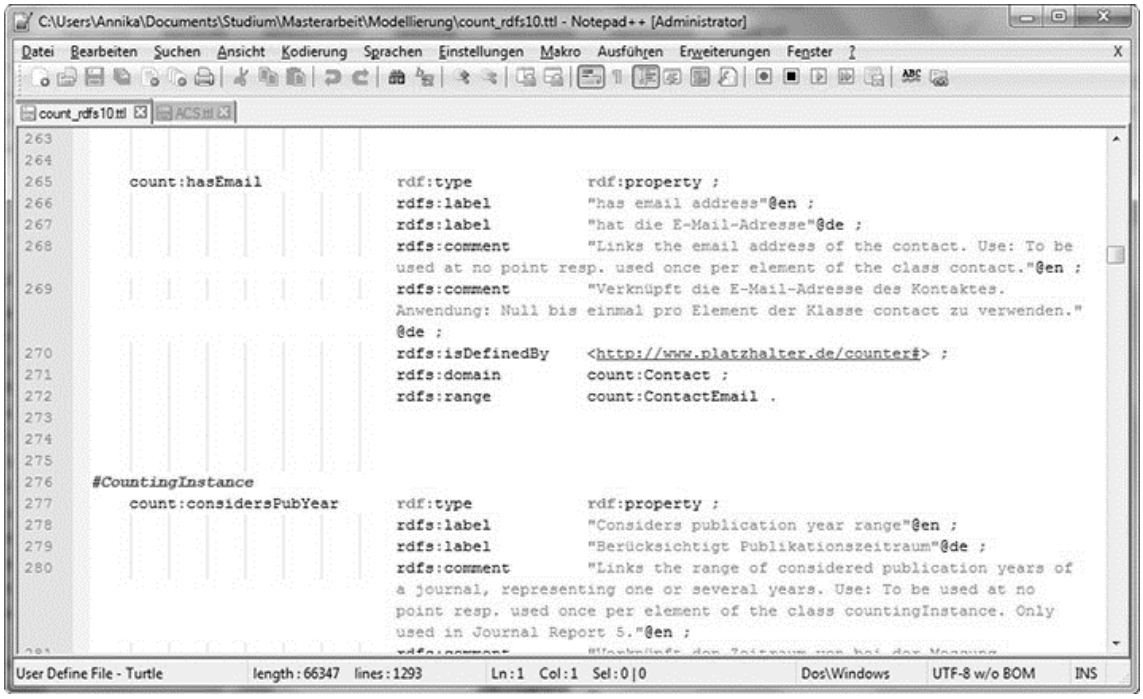


Abb. 13 *Screenshot, Notepad++ bei geöffneter Turtle-Datei.*

Besonders für das manuelle Erstellen von Turtle- aus XML-Dateien ist die Multi-Dokument-Ansicht mit Tab-Interface oder mehreren Fenstern sehr nützlich. Längere Dateien können außerdem in ein zweites, daneben angeordnetes Fenster dupliziert werden, wodurch ein effizientes Kopieren bzw. Umgestalten von Code möglich wird. Als letztes hervorzuhebendes Feature ist die umfangreiche Suchen- und Ersetzen-Funktion zu nennen. Einzelne Zeichenfolgen und auch weitere Elemente wie

[144] Vgl. Notepad++ Wiki - User Defined Language Files [Elektronische Ressource].

etwa Zeilenumbrüche können unter anderem gesucht, markiert und gezählt werden sowie innerhalb aller geöffneter Dateien, innerhalb des gesamten aktuellen Dokumentes oder auch nur innerhalb einer Markierung durch andere Ausrücke ersetzt werden. Neben diesen genannten Features bietet Notepad++ zahlreiche weitere hilfreiche Unterstützungen,[145] wie beispielsweise die Möglichkeit der Lesezeichensetzung im Code und eine anpassbare Benutzeroberfläche.

4.4.2 Raptor

Eine Syntax-Überprüfung des Turtle-Codes ist mit dem Parser/Serializer-Tool *Raptor (Raptor RDF Syntax Library)* möglich. Besonders für elementare Schreibfehler, wie etwa vergessene Punkte am Ende eines Tripels, bietet sich die Kontrolle mittels Raptor an.

Raptor ist Teil der von Dave Becket seit Anfang der 2000er Jahre entwickelten[146] Freeware-Suite *Redland RDF Libraries*, die verschiedene Bibliotheken für den Umgang mit RDF beinhaltet. Der Name der Software hat keine spezielle technische Bedeutung, sondern erinnert an Beckets früheren Arbeitsort, die *Redland area* um Bristol.[147] Zu der Suite gehört unter anderem die eigentliche *Redland RDF Library* mit der RDF-API und Triplestores, den RDF verwaltenden Datenbanksystemen. Zur korrekten Ausführung der Library werden außerdem *Raptor* und *Rasqal* benötigt. Die *Rasqal RDF Query Library* ist für die Ausführung von Suchanfragen zuständig, während die *Raptor RDF Syntax Library* zum Parsen und Serialisieren von RDF dient. Alle Bibliotheken sind in C geschrieben und

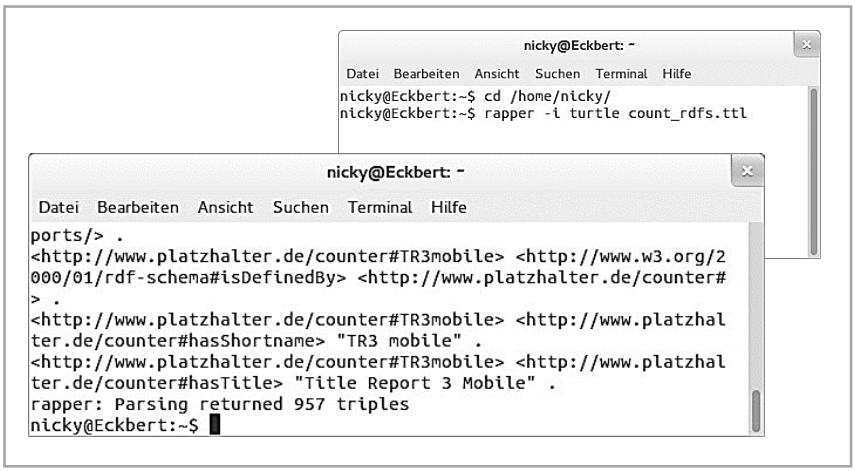


Abb. 14
Screenshots, Raptor im Terminal.

[145] Vgl. Notepad++ Features [Elektronische Ressource].
[146] Vgl. Librdf FAQ [Elektronische Ressource].
[147] Vgl. ebd.

werden theoretisch plattformübergreifend angeboten.[148] Die letzten Aktualisierungen der angebotenen Installationsdateien für Windows stammen allerdings aus dem Jahr 2006 und werden vom Entwickler selbst als veraltet bezeichnet,[149] es soll jedoch möglich sein, die Bibliotheken selbst für Windows und auch für Apples Mac OS X zu kompilieren.[150] Die aktuellsten Versionen für UNIX stammen aus dem Mai 2014[151] und wurden unter Ubuntu für diese Arbeit genutzt.

Die *Raptor RDF Syntax Library* ist auch einzeln und ohne graphische Oberfläche nutzbar und kann unter anderem dazu verwendet werden, die formale Korrektheit von Turtle-Syntax zu prüfen. Dazu wird im Terminal zunächst der Pfad der Turtle-Datei ausgewählt. Danach müssen das Inputformat Turtle sowie der Dateiname angegeben werden:

```
cd ~
rapper -i turtle example.ttl
```

Raptor extrahiert daraufhin die RDF-Tripel aus dem Turtle-Code. Wenn Fehler in der Syntax vorhanden sind, werden die Code-Zeile des Fehlers sowie die letzten fehlerfreien Tripel angegeben, was ein Auffinden des Fehlers im Code ermöglicht. Ist die Syntax korrekt, zeigt Raptor an, wie viele Tripel extrahiert wurden. Wird zusätzlich ein Output-Format wie etwa RDF/XML gewählt und ein Dateiname für die Output-Datei angegeben, serialisiert Raptor die zuvor extrahierten Tripel unter Verwendung der gewählten Sprache:

```
rapper -i turtle -o rdfxml example.ttl > example.rdf
```

4.4.3 OntoWiki

Eine weitere vorzustellende Software ist *OntoWiki*, die seit 2006 von der auch am AMSL-Projekt beteiligten Forschungsgruppe AKSW als OpenSource-Software entwickelt wird.[152] OntoWiki wurde zwar nicht direkt für die Modellierung benötigt, stellt aber das System dar, auf dem das ERMS nach Fertigstellung betrieben werden soll. Entsprechend wurden gewisse Besonderheiten von OntoWiki, wie etwa die Probleme bei der Verarbeitung von Blanknodes, bei der Modellierung zu berücksichtigen versucht.

[148] Vgl. Redland RDF Libraries [Elektronische Ressource].
[149] Vgl. Raptor Download Win32 [Elektronische Ressource].
[150] Vgl. Librdf FAQ [Elektronische Ressource].
[151] Vgl. Raptor RDF Syntax Library [Elektronische Ressource].
[152] Vgl. OntoWiki [Elektronische Ressource].

Der Name OntoWiki bezieht sich auf *semantische Wikis* (engl. *semantic wikis*), also Wikis, die ein semantisches Modell der in ihnen verzeichneten Daten enthalten. Dabei ist OntoWiki selbst ein nicht ganz typisches Semantic Wiki-System, da es den textuellen Inhalt des Wikis nicht in den Vordergrund stellt, sondern vorrangig als leicht bedienbare Schnittstelle zur gemeinschaftlichen Erstellung und Pflege von Wissensbasen genutzt werden kann.[153] Es geht nicht, wie andere semantische Wikis, von einem bereits bestehenden Wiki aus und reichert dieses mit RDF-Tripeln an, sondern fügt Wiki- und Community-Ansätze in das klassische Knowledge Engineering ein.[154]

OntoWiki ist also eine umfangreiche, web-basierte Software zum Management semantischer Daten. Als eine solche bietet es einen Ontologie-Editor, der sowohl Linked Data in allen üblichen Formaten wie Turtle oder RDF/XML einlesen kann als auch das Erstellen und Verwalten von Ressourcen, Verbindungen zwischen Ressourcen, Klassen und Ähnlichem in einer graphischen Oberfläche mit verschiedenen Ansichten ermöglicht. Diese ist sehr nutzerfreundlich gestaltet und auch von Anwendern mit wenig bis keinen Kenntnissen der Serialisierungssprachen bedienbar. Außerdem besitzt OntoWiki etliche Features zur Repräsentation der eigenen Daten im Semantic Web sowie weitreichende Möglichkeiten zur semantischen Suche, unter anderem über SPARQL. Es bietet sowohl einen Linked-Data-Server für die eigenen Daten als auch einen Client, um die eigenen durch Fremddaten aus dem Web zu ergänzen. Über externe Weiterverlinkungen der eigenen Daten kann informiert werden, bzw. können solche Daten gesendet werden. Da OntoWiki nicht von einem bestimmten Backend abhängig ist, können die eigentlichen Daten sowohl in einem Triplestore gespeichert sein, als auch an anderen Orten, wie etwa in einer konventionellen MySQL-Datenbank.[155] Die Community-Features bestehen insbesondere aus einer Änderungsverfolgung, Kommentar- und Annotationsfunktionen, Bewertungen der Popularität von Inhalten sowie einem Überblick über die Nutzeraktivität.

Genau die Einfachheit der Bedienung sowie die gute Unterstützung kooperativen Arbeitens machen OntoWiki zu dem Tool der Wahl für die Umsetzung des ERMS, das letztendlich von bibliothekarischen Anwendern aller sächsischen Hochschulen gemeinschaftlich genutzt werden soll. Nicht zuletzt qualifiziert sich die Software für das Projekt, da sie durch die beteiligte Forschungsgruppe AKSW entwickelt wurde und somit Kernkompetenzen für Funktionalitätsanpassungen im Team vorhanden sind.

---

[153] Vgl. Semantische Wikis [Elektronische Ressource], S. 11.
[154] Vgl. Dietzold, Sebastian: Koll[a]borative Wissensarbeit mit OntoWiki [Elektronische Ressource], S. 1.
[155] Vgl. What is OntoWiki? What can it do for you? [Elektronische Ressource].

5 Realisierung des RDF-Vokabulars

5.1 Grundlegende Modellierung: RDFS

Nachdem alle Grundlagen bekannt und die Vorarbeiten abgeschlossen sind, kann mit der eigentlichen Realisierung des Vokabulars begonnen werden. Als Erstes muss die eindeutige URI-Identifizierung der im Vokabular definierten Ressourcen sichergestellt werden. Dafür wird ein eigener Namespace angelegt. Da dessen endgültige URI zum Zeitpunkt der Erstellung dieser Arbeit allerdings noch nicht bekannt ist, wird im Turtle-Code ein Platzhalter verwendet, der wie folgt deklariert wird:

```
@prefix count: <http://www.platzhalter.de/counter#> .
```

Um sowohl den Fließtext als auch die Grafiken möglichst kompakt zu halten, wird in Qnames von nun an das Präfix `count:` eingekürzt. Steht also vor einem Identifikator lediglich ein Doppelpunkt, bezieht sich der QName auf den oben deklarierten COUNTER-Namespace.

Nun können die jetzt eindeutig identifizierbaren Ressourcen des Vokabulars definiert werden. Um die Übersichtlichkeit dieser Arbeit zu wahren, wird im Folgenden exemplarisch die Definition der Klasse `:Customer` und der in ihrer Umgebung befindlichen Ressourcen ausführlich erläutert. Es wird außerdem gezeigt, von welchen XML-Elementen die jeweiligen Ressourcen abgeleitet werden und welche inhaltlich-bibliothekarischen Besonderheiten bei der Definition und Nutzung der Ressource zu beachten sind. Diese hier beschriebenen Ressourcen sowie alle weiteren im Vokabular definierten Klassen und Eigenschaften werden in Anhang A übersichtlich dargestellt.

Die Klasse `:Customer` eignet sich besonders gut zur Vorstellung der verwendeten RDF- und RDFS-Ressourcen, da hier alle Besonderheiten auftreten, die bei der Modellierung zu beachten waren. Zunächst wird beschlossen, dass alle RDF-Ressourcen, die einen Kunden im Sinne des COUNTER-Reports darstellen, inhaltlich zusammengefasst werden sollen. Dies geschieht, indem sie alle als Instanzen der Klasse `:Customer` ausgezeichnet werden.

Entsprechend ist zunächst im Vokabular eben diese Klasse `:Customer` zu modellieren:

```
:Customer rdf:type rdfs:Class .
```

Die Eigenschaft `rdf:type` deklariert dabei eine Ressource als Instanz einer Klasse. Analog zu der hier genutzten Klasse `rdfs:Class`, der Klasse aller Klassen, existieren noch `rdf:Property`, `rdfs:Literal` und `rdfs:Resource` als Klassen aller Eigenschaften, Literale bzw. Ressourcen.[156] Letztere ist, da jedes von RDF beschriebene Element eine Ressource darstellt, die „class of everything“:[157] Alles ist eine Instanz der Klasse `rdfs:Resource`.[158]

Im ursprünglichen XML-Schema sind die komplexen Typen `c:Customer` und `c:Vendor` Erweiterungen des komplexen Typs `c:Organization`. Dabei wird über die Funktion `xs:extension`, die dem Ableiten neuer Typen durch Erweiterung eines vorhandenen Datentyps dient, dem Inhalt von `c:Customer` der Basiswert `c:Organization` zugeordnet. Dadurch sind alle den Typ `c:Organization` ausmachenden Inhalte auch in `c:Customer` enthalten. Dieser Basiswert wird um eine weitere Sequenz mit den beiden Elementen `c:Consortium` und `c:InstitutionalIdentifier` ergänzt.

Diese Konstruktion ist, zumindest angelehnt, auch auf die Linked-Data-Welt übertragbar. Es ist mit RDFS nicht nur, wie im Einführungskapitel gezeigt, möglich, Klassen zu definieren, sondern auch, diese Klassen hierarchisch miteinander zu verbinden. Dazu wird die Eigenschaft `rdfs:subClassOf` genutzt. Sie gibt an, dass alle Instanzen einer Klasse (Unterklasse) auch Instanzen einer anderen Klasse (Oberklasse) sind.[159] Im Falle der Konzepte von Datenanbieter, Kunde und Organisation kann ausgesagt werden, dass Datenanbieter und Kunde immer Organisationen sein sollen. Dies wird ausgedrückt durch

```
:Vendor rdfs:subClassOf :Organization .
:Customer rdfs:sublassOf :Organization .
```

Da `rdfs:Class` sowohl die Domain als auch die Range von `rdfs:sublassOf` ist,[160] ist die zu Anfang getätigte Aussage

```
:Customer rdf:type rdfs:Class .
```

in den obigen Aussagen bereits enthalten. Der Übersichtlichkeit und Verständlichkeit des Codes halber bleibt dieses eigentlich überflüssige Tripel aber trotzdem in der Definition von `:Customer` enthalten.

[156] Vgl. RDF 1.1 Schema Recommendation [Elektronische Ressource].
[157] RDF 1.1 Schema Recommendation [Elektronische Ressource].
[158] Vgl. RDF 1.1 Schema Recommendation [Elektronische Ressource].
[159] Vgl. ebd.
[160] Vgl. ebd.

Um in Anzeigesoftware wie etwa OntoWiki eine bessere Übersichtlichkeit zu erreichen, ist es möglich, mit Hilfe der Eigenschaft `rdfs:label` einem Individuum einen menschenlesbaren Namen in Form eines Literals zuzuordnen. Mehrsprachige Labels werden dabei durch die Möglichkeit des *Language Taggings* von RDF-Literalen unterstützt.[161] Diese Tags, ihr Aufbau und ihre Nutzung wurden von der IETF (*Internet Engineering Task Force*, Teil der *Internet Society ISOC*) als *Best Current Practise* dokumentiert.[162] Im Turtle-Code ergeben sich für den Kunden folgende Tripel:

```
:Customer rdfs:label "customer"@en, "Kunde"@de .
```

Ähnlichen Zwecken wie das Label dient der Kommentar: Die Eigenschaft `rdfs:comment` verknüpft ein Individuum mit einer menschenlesbaren Beschreibung, die die Bedeutung von Klassen und Eigenschaften erklärt. Wie auch `rdfs:label` besitzt `rdfs:comment` die Range `rdfs:Literal`, weshalb ein Kommentar immer ein Literal ist. Auch er kann mit Language Tags versehen werden.[163]

```
:Customer rdfs:comment "Details on the customer whose data the
                        report represents."@en, "Details zu dem
                        Kunden, über dessen Daten der Report
                        berichtet."@de .
```

Hier kommt bereits zum Ausdruck, dass eine Instanz der Klasse `:Customer` selbst noch keine konkreten Informationen wie etwa den Namen des Kunden enthält. Sie ist aber eine RDF-Ressource, mit der Eigenschaften verknüpft werden können. Diese wiederum verbinden dann einzelne, die Fakten beinhaltenden Ressourcen. Dies hätte theoretisch auch über Blanknodes realisiert werden können, ohne dass eine zusammenfassende Ressource der Klasse `:Customer` nötig wäre. Allerdings wird wegen der Verarbeitungsprobleme von OntoWiki eine Vermeidung von Blanknodes angestrebt. Entsprechend werden die Klassen und Eigenschaften so modelliert, dass keine Blanknodes verwendet werden müssen.

Zunächst sollen aber noch mit Hilfe der Eigenschaft `rdfs:seeAlso` bzw. mit Hilfe ihrer Untereigenschaft `rdfs:isDefinedBy` weiterführende Informationen über die Subjektressource angegeben werden.[164] Die Eigenschaft `rdfs:isDefinedBy` stellt hier den speziellen Fall der Angabe des Definitionsortes der Subjektressource dar. Hier ist dies die Definition des COUNTER-Vokabulars:

```
:Customer rdfs:isDefinedBy <http://www.platzhalter.de/counter#> .
```

[161] Vgl. RDF 1.1 Schema Recommendation [Elektronische Ressource].
[162] erreichbar unter http://tools.ietf.org/html/bcp47
[163] Vgl. RDF 1.1 Schema Recommendation [Elektronische Ressource].
[164] Vgl. ebd.

Damit ist die Definition der Klasse `:Customer` abgeschlossen, insgesamt ergeben sich durch folgenden Code sieben Tripel:

```
:Customer
       rdf:type rdfs:Class ;
       rdfs:subClassOf count:Organization ;
       rdfs:label "customer"@en, "Kunde"@de ;
       rdfs:comment "Class of customers whose data the reports
           represent. Only organizations can be customers."@en, "Klasse
           der Kunden, über deren Daten die Reports berichten. Nur
           Organisationen können Kunden sein "@de ;
       rdfs:isDefinedBy <http://www.platzhalter.de/counter#> .
```

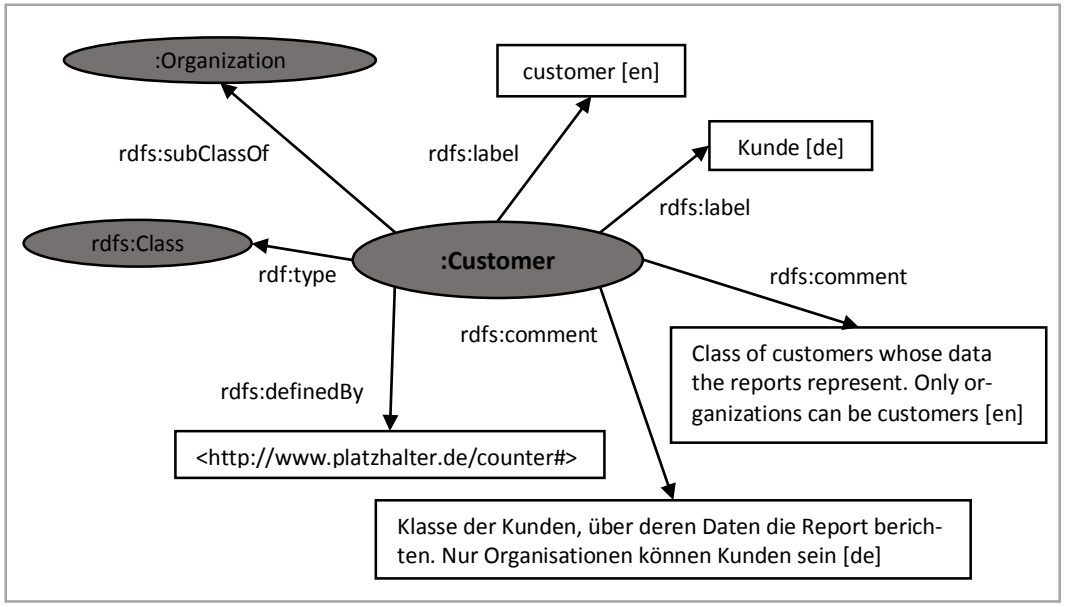


Abb. 15 *Alle bisher definierten Ressourcen im Umfeld der Klasse* `:Customer`.

Die oben erwähnten weiteren Ressourcen, die die eigentlichen Fakten beinhalten sollen, werden teilweise ebenfalls zu Klassen zusammengefasst. Die Entscheidung, für welche Ressourcen solch eine Zusammenfassung sinnvoll ist, ist nicht immer eindeutig und objektiv zu fällen. Obwohl Linked-Data-Vokabulare grundsätzlich auch die Möglichkeit der Nachnutzung durch andere Modellierer berücksichtigen sollen und damit nicht zu eng auf den konkret geplanten Einsatz zuzuschneiden sind,[165] soll hier nur dann eine Klasse gebildet werden, wenn ein konkreter Nutzen daraus denkbar ist. So kann es beispielsweise nützlich sein, durch eine einfache Abfrage alle Ressourcen ausgeben zu lassen, die einen Kontakt bzw. eine Kontaktperson darstellen, um etwa eine wichtige Änderung an all diese Personen kommunizieren zu können. Folglich ergibt eine Klasse `:Contact`

[165] Vgl. Allemang, Dean: Semantic web for the working ontologist, S. 311 f.

Sinn, da diese die genannte Abfrage vereinfachen würde. Ein konkreter Nutzen der Zusammenfassung jeglicher Zählstände dagegen scheint nicht klar erkennbar, weshalb keine Klasse für die jene Ressourcen gebildet wird, welche Zählstände darstellen. Analog war auch bei allen anderen Ressourcen für bzw. gegen die Modellierung von Klassen zu entscheiden. Sollte sich im Laufe der praktischen Anwendung des Vokabulars aber doch ein Problem ergeben, das durch die Modellierung einer Klasse einfacher zu lösen wäre, kann das Vokabular jederzeit um eine solche ergänzt werden.

Wie soeben festgestellt, lohnt sich die Modellierung einer Klasse `:Contact`. Dies geschieht analog zu der oben geschilderten Definition der Klasse `:Customer`:

```
count:Contact
       rdf:type rdfs:Class ;
       rdfs:label "contact"@en, "Kontakt"@de ;
       rdfs:comment " Class of contacts"@en, " Klasse der
           Kontaktmöglichkeiten"@de ;
       rdfs:isDefinedBy <http://www.platzhalter.de/counter#> .
```

Ebenso werden die Klassen `:Organization`, `:Consortium` und `:Report` definiert. Diese sind, wie an Abbildung 16 zu erkennen, bisher größtenteils unverbunden und stehen noch für sich allein.

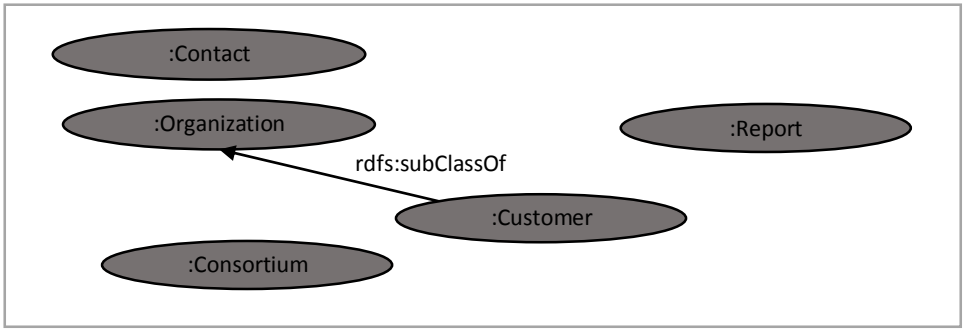


Abb. 16 *Die bisher definierten Klassen im Umfeld der Klasse `:Customer` und deren Beziehungen untereinander.*

Etliche der zu Anfang der Modellierung der Klasse `:Customer` im XML-Schema analysierten Elemente sind bisher noch nicht wieder aufgegriffen worden. Das liegt daran, dass sie später als Ressourcen modelliert werden, die in keiner speziellen COUNTER-Klasse zusammengefasst werden. Trotzdem werden sie mit bestimmten Eigenschaften verknüpft, ebenso jene Ressourcen, die Instanzen einer COUNTER-Klasse sind. Entsprechend gehen die Datentypen des XML-Schemas nicht vorrangig in RDFS-Klassen auf, sondern in Eigenschaften.

Diese sind noch zu definieren, was zunächst am Beispiel der Beziehung zwischen einem Report und dem diesen erhaltenden Kunden gezeigt werden soll, welche nicht im XML-Schema vorgesehen ist. Diese Beziehung dient dazu, Report und Kunde einander dauerhaft zuordnen zu können. Die entsprechende Ressource soll den Namen `:receives` erhalten und ist eine Instanz der Klasse aller Eigenschaften:

```
:receives rdf:type rdfs:Property .
```

Sie erhält, genau wie zuvor bei der Definition von Klassen gesehen, sowohl einen menschenlesbaren Namen als auch eine solche Beschreibung, beides sowohl auf Deutsch als auch auf Englisch. Außerdem wird angegeben, wo die Ressource definiert wird:

```
:receives
       rdfs:label "receives"@en, "erhält"@de ;
       rdfs:comment "Links the COUNTER report which the customer
           receives."@en, "Verknüpft den COUNTER
           Report, den der Kunde erhält."@de ;
       rdfs:isDefinedBy <http://www.platzhalter.de/counter#> .
```

Um festzulegen, dass jedes Subjekt eines Tripels mit der Eigenschaft `:receives` als Prädikat eine Instanz der Klasse `:Customer` ist, wird letztere als Domain von `:receives` festgelegt. Analog wird, durch Festlegung der Range, jedes Objekt eines solchen Tripels zur Instanz der Klasse `:Report` erklärt:

```
:receives
       rdfs:domain :Customer ;
       rdfs:range :Report .
```

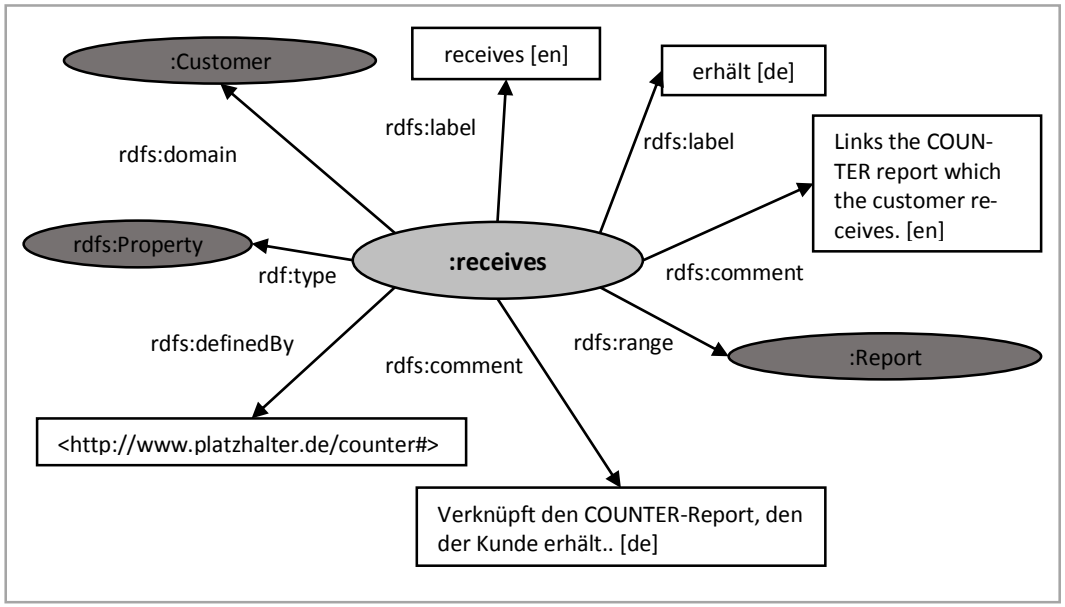


Abb. 17 *Die bisher definierten Ressourcen im Umfeld der Eigenschaft* `:receives`.

Im XML-Schema waren, wie zu Anfang gesehen, noch weitere, den Kunden beschreibende Elemente aufgeführt. Diese gehen in ebenfalls in Eigenschaften auf. Analog zu der Definition von :receives werden die Eigenschaften :hasName für den Organisationsnamen, :hasOrganizationID für die vom Datenanbieter zugeordnete ID der Organisation, :hasContact zur Verknüpfung der beschriebenen Klasse :Contact und :hasWebsiteURL sowie :hasLogoURL für die entsprechenden Web-Adressen definiert.

Ähnlich wie im XML-Schema, wo diese Elemente zum komplexen Typ c:Organization gehören, spielt auch im RDF-Vokabular die Klasse :Organization eine Rolle: Die Domain der aufgezählten Eigenschaften ist nicht :Customer, sondern :Organization. Dadurch können diese Properties auch mit einem Subjekt der Klasse :Vendor genutzt werden, da diese genau wie :Customer eine Unterklasse von :Organization ist.

Die Definition einer solchen Eigenschaft ist hier am Beispiel von :hasWebsiteURL dargestellt.

```
count:hasWebsiteURL
       rdf:type rdf:Property ;
       rdfs:label "has the website url"@en, "hat die
           Website-URL"@de ;
       rdfs:comment "Links the URL of an organization
           website."@en, "Verknüpft die URL einer Organisations-
           Website."@de ;
       rdfs:isDefinedBy <http://www.platzhalter.de/counter#> ;
       rdfs:domain count:Organization .
```

Die zwei Eigenschaften :isPartOfConsortium und :hasInstitutionalIdentifer haben wie :receives die Domain :Customer, da die ihnen im XML-Schema entsprechenden Elemente nur für den Gebrauch im Datentyp c:Customer vorgesehen sind. Die Eigenschaft :hasInstitutionalIdentifier ist noch lediglich ein Platzhalter, da inhaltlich laut COUNTER-Schema bisher keine zulässigen Identifier in Gebrauch sind.

Insgesamt ist nun bereits ein recht umfangreiches Netz an Klassen und Eigenschaften definiert, das in Abbildung 18 graphisch dargestellt wird.

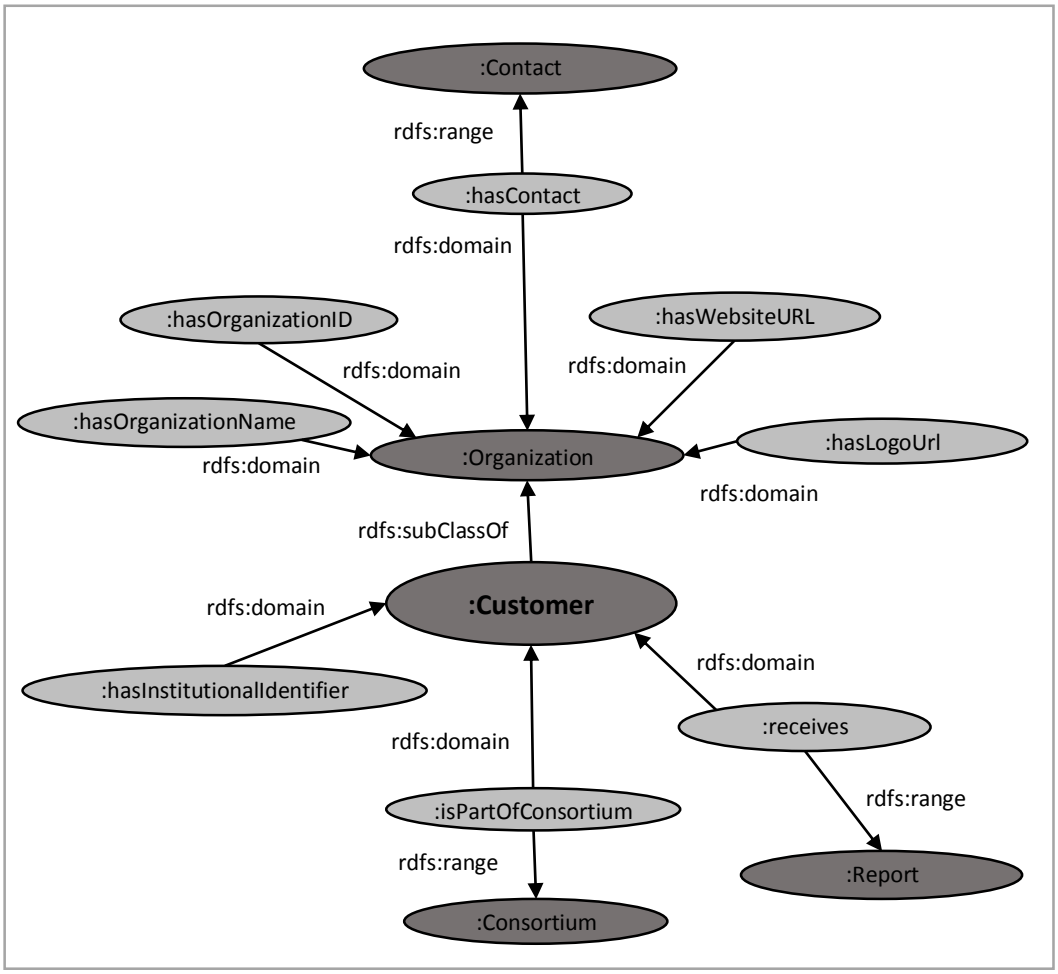


Abb. 18 *Die bisher definierten Klassen und Eigenschaften im Umfeld der Klasse* `:Customer` *und deren Beziehungen untereinander.*

Eine modellierte Eigenschaft sowie ein RDF-Modellierungskonzept sollen zusätzlich zu diesem Beispiel noch hervorgehoben werden, da sie nicht im XML-Schema vorgesehen sind, aber in den COUNTER-RDF-Reports eine wichtige Position einnehmen. Wie bereits in der Grundkonzeption erwähnt, werden bei über mehrere Reports hinweg auftretenden, aber prinzipiell veränderlichen Daten sowohl die neuen als auch die alten Informationen zugeordnet und aufgehoben. Es wird also eine Art Historie angelegt. Dies geschieht mit Hilfe des Konzeptes der *Reifikation* (wörtlich: *Vergegenständlichung*), welches es ermöglicht, Aussagen über Aussagen zu treffen, sowie mit Hilfe der Eigenschaft `:sourceReport`, die diese getroffene Aussage mit ihrem Quellreport verbindet.

Die Reifikation dient dazu, eine gesamte Aussage als Subjekt oder Objekt einer anderen verwenden zu können. Die Aussage selbst wird also vergegenständlicht, um über diesen Gegenstand neue Aussagen zu treffen. Damit können Informationen wie *„Wikipedia sagt, dass Shakespeare im Jahr 1564 geboren wurde.“* oder eben auch *„Die Zeitschrift ist laut Report x über die Platform y zugänglich.“* dargestellt werden. In einer solche Tripel auswertenden Software kann festgelegt werden, ob der

Aussage von Wikipedia zu trauen ist bzw. der Report x die gewünschte Quelle darstellt - oder nicht. In letzterem Fall werden die Aussagen über Shakespeares Geburtsjahr und die Zugangsplattform ignoriert, in ersterem als gültig anerkannt.

Syntaktisch wird die Reifikation bereits von RDF unterstützt, wo die Eigenschaften `rdf:subject`, `rdf:predicate` und `rdf:object` für die einzelnen Teile sowie die Klasse `rdf:statement` für die gesamte reifizierte Aussage angelegt sind. Am Beispiel der Plattform einer Zeitschrift sieht eine Reifikation wie folgt aus:

```
ex:n1 a rdf:statement ;
     rdf:subject ex:Journal_1 ;
     rdf:predicate count:isAccessibleVia ;
     rdf:object ex:Platform_1 .
```

Die Verknüpfung mit dem die Aussage treffenden Quellreport geschieht dann über die Eigenschaft `count:sourceReport`:

```
ex:n1 count:sourceReport ex:Report_1 .
```

Ändert sich nun die Zugangsplattform für diese eine Zeitschrift, wird einfach eine zweite reifizierte Aussage über die neue Plattform getroffen und ihrerseits mit ihrem Quellreport verknüpft:

```
ex:n2 a rdf:statement ;
    rdf:subject ex:Journal_1 ;
    rdf:predicate count:isAccessibleVia ;
    rdf:object ex:Platform_2 ;
    count:sourceReport ex:Report_2 .
```

Graphisch dargestellt sind die beiden reifizierten Aussagen in der umseitigen Abbildung.

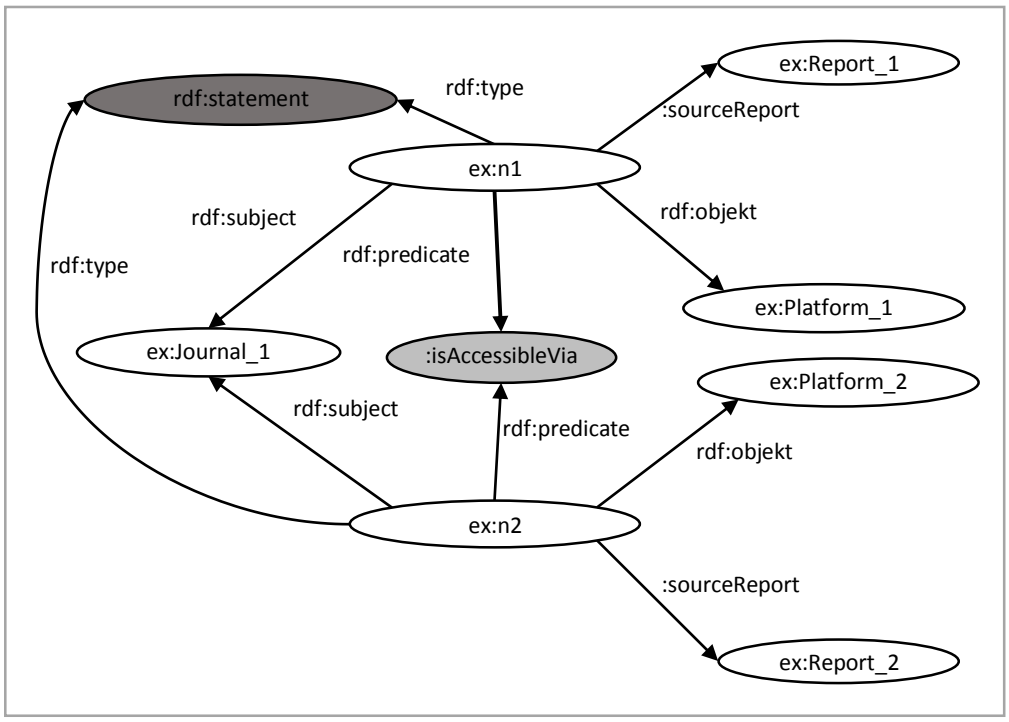


Abb. 19 *RDF-Beispiel für eine Reifikation und die Eigenschaft* `:sourceReport` *bei einer veränderten Zugangsplattform.*

5.2 Erweiterung: OWL

Durch die bisherige RDFS-Modellierung ist es möglich, alle in den XML-Reports enthaltenen Informationen in RDF auszudrücken und sinnvoll miteinander zu verbinden. Allerdings sind dadurch noch nicht alle Informationen des XML-Schemas im Vokabular enthalten. Wie in der Anforderungsanalyse festgestellt, könnte eine Umsetzung möglichst aller Informationen des XML-Schemas für potentielle andere Nutzer des Vokabulars hilfreich sein. Sollten etwa zukünftig Reports bereits nativ im RDF-Format erstellt werden, wäre es nützlich, die Einschränkungen des XML-Schemas in das Vokabular zu übertragen. In Kapitel *4.2 Analyse des COUNTER XML-Schemas* war zu sehen, dass im Schema Einschränkungen für Datentypen, zulässige Werte von Elementen und Kardinalitäten von Elementen getroffen werden. Für die Umsetzung all dieser Einschränkungen ist RDFS nicht mehr mächtig genug und es wird nötig, auf OWL zurückzugreifen.

Generell ist es allerdings auch in OWL nicht möglich, das dargestellte Wissen einzuschränken. OWL dient als Ontologiesprache dazu, aus bestehendem Wissen neues Wissen zu folgern. Es ist also

keine Schemasprache für Syntaxkonformität, wie es bei XML-Schema der Fall ist.[166] Daraus resultiert ganz prinzipiell die Unmöglichkeit, die Einschränkungen aus dem XML-Schema unverändert in das Vokabular zu übernehmen. Einige von OWL angebotene Konstrukte sind aber in der Lage, Festlegungen bezüglich Datentypen, Elementen von Klassen oder auch Kardinalitäten zu treffen.[167] Diese dienen allerdings niemals der Einschränkung, sondern immer der Folgerung neuen Wissens. Welche Konsequenzen daraus erwachsen, wird in den folgenden Beschreibungen der einzelnen Konstrukte einzeln thematisiert.

Zunächst soll die Einschränkung von Elementen auf bestimmte Datentypen aus dem XML-Schema betrachtet werden. Im Vokabular haben die Eigenschaften die Aufgabe, RDF-Ressourcen miteinander zu verknüpfen und ordnen diese, falls solche modelliert wurden, über Domain und Range den entsprechenden Klassen zu. Etliche Eigenschaften, wie etwa `:hasCount` im Umfeld von Zählinstanzen oder `:hasStartDay` im Umfeld von Zeiträumen, verknüpfen als Objektressourcen keine Individuen mit URI-Referenzen, sondern Literale. Für die Daten, die in den Literalen gespeichert werden, sind im XML-Schema oftmals bestimmte Datentypen definiert. Das Kernproblem ist also, für eine Eigenschaft festzulegen, dass ihre Objektressourcen Literale eines bestimmten Datentyps sind.

Dafür bietet OWL zunächst eine Unterscheidung von Eigenschaften an. Während diese in RDFS immer als Property deklariert werden, differenziert OWL in *Object Properties* und *Datatype Properties*. Während erstere zwei Individuen verknüpfen, verbinden Datatype Properties ein Individuum mit einem Datenwert, also einem Literal.[168] Entsprechend werden die in RDFS bereits als `rdf:Property` deklarierten Datatype Properties im Code zunächst entsprechend umdeklariert,[169] wobei stets zuvor geklärt sein muss, ob die zu verbindenden Ressourcen tatsächlich immer Literale bzw. bei Object Properties immer Individuen darstellen werden.

```
~~:hasCount rdf:type rdf:Property .~~
~~:hasStartDay rdf:type rdf:Property .~~
```

```
:hasCount rdf:type owl:DatatypeProperty .
:hasStartDay rdf:type owl:DatatypeProperty .
```

[166] Vgl. OWL 2 Web Ontology Language Primer (Second Edition), W3C Recommendation [Elektronische Ressource].
[167] Vgl. ebd.
[168] Vgl. ebd.
[169] Vgl. ebd.

Nun kann die eigentliche Definition des Datentyps der Objektressource erfolgen. Dies geschieht über den schon bekannten Domain/Range-Mechanismus, allerdings mit dem Unterschied, dass mittels Range der Objektressource nun keine gewöhnliche Klasse, sondern ein Datentyp bestimmt wird. Eine Auflistung der für eine solche Verknüpfung empfohlenen, in OWL integrierten XML-Datentypen ist beim W3C einzusehen.[170] Es ergibt sich nach Deklaration des noch fehlenden XML-Schema-Namensraumes folgender Turtle-Code für die beiden Beispieleigenschaften:

```
@prefix xsd: <http://www.w3c.org/2001/XMLSchema#> .

:hasStartDay
       rdf:type owl:DatatypeProperty ;
       rdfs:range xsd:date .
:hasCount
       rdf:type owl:DatatypeProperty ;
       rdfs:range xsd:nonNegativeInteger .
```

Wie oben bereits erläutert, ist dies keine Einschränkung im Sinne von XML-Schema. Anstatt als ungültig eingestuft zu werden, wird bei einer Verknüpfung von Objektressourcen eines abweichenden Datentyps sehr wohl ein gültiges Tripel erstellt. Es werden lediglich objektiv falsche Schlussfolgerungen gezogen, sodass das dargestellte Wissen in sich schlüssig und korrekt bleibt. Die Modellierung dient in diesem Fall also eher als Hinweis für andere Modellierer sowie als weitere semantische Anreicherung des im Vokabular repräsentierten Wissens.

Als nächstes soll das Konzept einer eingeschränkten Anzahl zulässiger Werte für XML-Elemente, wie es an einigen Stellen im XML-Schema definiert wird, auf das Vokabular übertragen werden. OWL bietet hier das Konzept einer geschlossenen Klasse, deren Instanzen abschließend aufgezählt werden:[171] Beispielsweise darf die Klasse `:Category` nur die Instanzen haben darf, die in der Datei `counterElements4_0.xsd` angegebenen werden. Sie soll daher mit Hilfe von OWL über die Eigenschaft `owl:oneOf` als geschlossene Klasse modelliert werden. Zunächst wird die bisherige Deklaration als RDFS-Klasse zu einer OWL-Klasse abgeändert. Dies ist prinzipiell nicht nötig, da durch die nachfolgende Verwendung einer OWL-Einschränkung die Klasse automatisch als OWL-Klasse angesehen wird. Im Sinne der besseren Übersichtlichkeit des Turtle-Code empfiehlt sich die Änderung dennoch:

```
~~:Category rdf:type rdfs:Class .~~
:Category rdf:type owl:Class .
```

[170] Erreichbar unter http://www.w3.org/TR/2012/REC-owl2-quick-reference-20121211/#Built-in_Datatypes_and_Facets.
[171] Vgl. OWL 2 Web Ontology Language Primer (Second Edition), W3C Recommendation [Elektronische Ressource].

Nun folgt die eigentliche Einschränkung, die durch die Angabe einer äquivalenten Klasse geschieht. Dabei wird die Klasse :Category nicht direkt durch ihre aufgezählten Instanzen definiert. Stattdessen wird ausgesagt, dass eine geschlossene OWL-Klasse mit den aufgezählten Instanzen existiert - und dass :Category zu dieser anderen OWL-Klasse äquivalent ist. Dieses Konstrukt hilft bei der Strukturierung und trägt zur Übersichtlichkeit bei:

```
:Category
       owl:equivalentClass [
           rdf:type owl:Class ;
            owl:oneOf (
                :Requests
                :Searches
                :AccessDenied
             )
        ] .
```

Dies bedeutet nun, dass die Klasse :Category keine anderen Elemente haben kann, als die aufgezählten. Allerdings wieder nicht im Sinne einer klassischen Einschränkung: Sollte nun beispielsweise in einem anderen Tripel definiert werden, dass ex:CategoryX eine Instanz von :Category ist, wird geschlossen, dass ex:CategoryX mit einer der aufgezählten Instanzen identisch sein muss.[172]

An diesem Beispiel wird deutlich, dass noch weitere Definitionen notwendig sind: Um Instanzen einer Klasse aufzählen zu können, sollten diese auch korrekt definiert werden. Deshalb umfasst das COUNTER-Vokabular neben definierten Klassen und Eigenschaften auch definierte Individuen. Ihnen werden auch erklärende Labels und Kommentare beigefügt, ebenso Siehe-auch-Verweise und ähnliches. Als Beispiel dient die folgende Definition der Ressource :Requests im COUNTER-Namespace:

```
count:Requests
       rdf:type count:Category ;
       rdfs:label "requests"@en, "Anfragen"@de ;
       rdfs:comment "example"@en, "Beispiel"@de .
```

Schließlich bleibt die Einschränkung der Kardinalität der XML-Elemente umzusetzen. So ist etwa für das XML-Element c:Category festgelegt, dass es genau einmal pro Zähleinheit zu verwenden ist, während eine Website-URL für Organisationen nicht verpflichtend, aber höchstens einmal anzuge-

[172] Vgl. OWL 2 Web Ontology Language Primer (Second Edition), W3C Recommendation [Elektronische Ressource].

ben ist. Insgesamt sind vier Varianten möglich: verpflichtend und unbegrenzt häufig, nicht verpflichtend und unbegrenzt häufig, nicht verpflichtend und höchstens einmal auftretend, verpflichtend und höchstens einmal, also genau einmal auftretend.

Über das Konstrukt der *Cardinality Restriction* lassen sich eine exakte Anzahl sowie Mindest- und Maximalwerte für die Verknüpfung von Eigenschaften angeben.[173] Für die Beispiele der genau einen zulässigen Kategorie pro Zählinstanz und der gar nicht oder nur einmal pro Organisation anzugebenen Website-URL ergibt sich folgender Turtle-Code:

```
CountingInstance
       rdfs:subClassOf [rdf:type owl:restriction ;
                        owl:onProperty :hasCategory ;
                        owl:cardinality
                          "1"^^xsd:nonNegativeInteger
                        ] .

Organization
       rdfs:subClassOf [rdf:type owl:restriction ;
                        owl:onProperty :hasWebsiteURL ;
                        owl:minCardinality
                          "0"^^xsd:nonNegativeInteger ;
                        owl:maxCardinality
                          "1"^^xsd:nonNegativeInteger
                        ] .
```

Damit wurde jedoch auch hier nicht der effektive Einsatz der Eigenschaften beschränkt, sondern lediglich die Klassen genauer definiert. :CountingInstance ist nun eine Unterklasse der Klasse all jener Ressourcen, die genau eine COUNTER-Kategorie besitzen. :Organization ist eine Unterklasse der Klasse aller Ressourcen, die eine oder keine Website-URL besitzen.[174] Es wird also ausgesagt, dass nur solche Ressourcen Instanzen der Klasse der Zähleinheiten sind, die genau eine Kategorie haben und nur solche Ressourcen Instanzen der Klasse der Organisationen, die immer entweder keine oder nur eine Website-URL besitzen. Daraus folgt, dass für den Fall einer Ressource ohne Kategorie, die dennoch als Instanz von :CountingInstance definiert wird, entsprechend der Open World Assumption geschlossen wird, dass die Ressource eine momentan lediglich nicht bekannte Kategorie besitzt. Werden anders herum zwei Kategorien zugeordnet, wird geschlossen, dass diese identisch sein müssen.

[173] Vgl. OWL Web Ontology Language Reference, W3C Recommendation [Elektronische Ressource].
[174] Vgl. ebd.

Bei der Angabe einer minimalen oder exakten Kardinalität ist im Vokabular weiterhin zu beachten, dass etliche Sachverhalte der organisations- und ressourcenspezifischen Art über Reifikationen modelliert werden sollen. Da in diesem Fall die Ressource nicht direkt mit der Eigenschaft verbunden wird, sondern lediglich als Subjekt einer reifizierten Aussage mit entsprechendem Prädikat ausgewiesen wird, wird dies nicht als direkte Verwendung der Eigenschaft erkannt. Beispielsweise wird im XML-Schema die Angabe exakt einer Plattform pro Reportelement gefordert. Würde nun für die Zugehörigkeit zu der Klasse `:ReportItem` genau eine Verwendung der Eigenschaft `:IsAccessibleVia` gefordert, wäre keine Plattform-Ressource Instanz von `:Platform`. Denn die Plattform wird nur über eine Reifikation angegeben, `:IsAccessibleVia` ist in diesem Fall also nur das Objekt der Eigenschaft `rdf:Property`. Folglich werden für Eigenschaften, die nur in reifizierten Aussagen verwendet werden sollen, keine Einschränkungen bezüglich der Kardinalität gemacht.

5.3 Übertragung von XML-Daten nach RDF

Nachdem das eigentliche Vokabular modelliert wurde, ist es notwendig, die Überführung der Daten aus XML-Reports in das RDF-Format zu betrachten. Dabei werden aus den Werten der XML-Elemente RDF-Ressourcen gebildet, die unter anderem über im Vokabular definierte Eigenschaften miteinander verbunden und den dort definierten Klassen zugeordnet werden.

Diese Konvertierung soll im funktionsfertigen System automatisiert mit Hilfe eines Konverter-Tools geschehen. Unter anderem um dessen Programmierung zu erleichtern, wird in Anhang C der Aufbau einer kompletten Reportdatei sequentiell durchgegangen und die Informationen elementweise in RDF-Daten überführt. Dabei wird die Turtle-Serialisierung genutzt. Hier soll ein Ausschnitt des `c:Vendor`-Elements als Beispiel dienen, um die Übertragung zu veranschaulichen.

Zunächst muss entschieden werden, ob eine Information als Literal oder als Individuum modelliert werden soll. In einem Tripel als Subjekt verwendete Ressourcen müssen grundsätzlich Individuen sein. Bei Objekt-Ressourcen in Tripeln bedingt sich die Wahl durch die als Prädikat verwendete Eigenschaft: ist sie eine Datatype Property, muss ein Literal, ist sie eine Object Property, ein Individuum verbunden werden. Handelt es sich um eine Eigenschaft, die nur als `rdfs:Property` definiert ist, muss selbst entschieden werden, ob das Objekt im späteren Kontext auch Subjekt eines anderen

Tripels werden soll oder ob ein Literal für die Modellierung ausreicht. Solche Eigenschaften sind im COUNTER-Vokabular allerdings nicht vorhanden.

Individuen müssen stets über einen URI verfügen, der sie eindeutig identifiziert. Diese Thematik ist recht umfangreich und wird deshalb in dem nachfolgenden Kapitel *5.4 URI-Vergabe* gesondert betrachtet. An dieser Stelle sollen zunächst Platzhalter verwendet werden. Der Namensraum der URIs von aus den Reportdaten erstellten Ressourcen soll nicht mit dem des Vokabulars übereinstimmen, um dieses übersichtlich zu halten und nicht mit Hunderten von Ressourcen zu überschwemmen. Stattdessen wird ein zweiter, ebenfalls als Platzhalter angelegter Namensraum mit dem Präfix `data:` deklariert:

```
@prefix data: <http://www.platzhalter.de/erm/data#> .
```

Das betrachtete Beispiel beginnt mit dem Vendor-Tag. Tritt dieser auf, steht bereits fest, dass eine Vendor-Ressource modelliert werden muss. Diese kommt häufig als Subjekt vor und muss daher ein Individuum sein. Sie kann der Klasse `:Vendor` zugeordnet werden. Dies ist theoretisch nicht nötig, da im weiteren Verlauf Eigenschaften mit dieser Ressource verbunden werden, die die Ressource mittels ihrer Range als eine Instanz der Klasse `:Vendor` definieren. Allerdings ist eine ausdrückliche Klassenzuweisung eindeutiger und direkter, was sowohl beim Lesen des Codes als auch bei der Verarbeitung durch Software nützlich sein kann. Ressourcen werden also nachfolgend immer direkt als Instanz einer Klasse deklariert, auch wenn sich diese Zuordnung durch Domain oder Range indirekt ebenfalls ergibt.

Da man außerdem weiß, dass der im Report genannte Datenanbieter diesen Report immer erstellt hat, kann diese Beziehung ebenfalls bereits hier mit Hilfe der Object Property `:creates` ausgedrückt werden.

`<Vendor>`	`data:Vndr_x a :Vendor .` `data:Vndr_x :creates data:Rprt_x.`

Das erste Unterelement von Vendor ist das Element Name, das einen Wert des Datentyps `xsd:string` enthält. Dieser wird zunächst genutzt, um der Ressource `data:Vndr_x` über `rdfs:label` eine menschenlesbare Bezeichnung zu geben. Diese soll aus eben jenem Wert sowie dem Spezifikator `"[Vendor]"` bestehen, da teilweise die gleichen Bezeichnungen für unterschiedliche Funktionen stehen, der Name des Datenanbieters beispielsweise auch der des Verlegers sein kann. Da `rdfs:label` auf Literale verweist, wird hier ein solches modelliert. Wäre der Wert von *strg1* bei-

spielsweise "0vid", lautete das Label "0vid [Vendor]". Die Verwendung des Namens des Datenanbieters für das Label seiner Ressource muss kritisch gesehen werden und sollte noch überdacht werden. Bei einer Namensänderung würde dadurch ein zweites Label mit dem neuen Namen angelegt, das nicht von dem alten zu unterscheiden ist.[175] Eine Reifikation bei einem Label ist allerdings wegen der geplanten Verwendung von OntoWiki ebenfalls nicht möglich, da die Software ein solches Label nicht anzeigt.

<Name> *strg5* </Name>	data:Vndr_x rdfs:label "*strg5* [Vendor]". …

Neben der Verwendung als Label ist der Organisationsname natürlich auch als solcher selbst anzugeben. Dies kann über die Datatype Property :hasOrganisationName umgesetzt werden, die ein Literal des Datentyps verknüpft. Da es aber wie oben beschrieben zu Veränderungen kommen kann, wird hier eine Reifikation modelliert, also eine Aussage über eine Aussage getroffen. Damit kann ausgedrückt werden, dass die Organisation diesen Namen laut dem Report Rprt_x besitzt. Somit ist eine Unterscheidung eventueller doppelter Namen möglich. Für die Reifikation wird eine Ressource data:n1 als rdf:statement modelliert, also als Aussage, die Subjekt, Prädikat und Objekt besitzt. Als diese werden nun die entsprechenden Ressourcen verbunden, die die Aussage über den Organisationsnamen bilden.

(Fortsetzung)	data:n1 a rdf:statement . data:n1 rdf:subject data:Vndr_x . data:n1 rdf:predicate :hasOrganizationName . data:n1 rdf:object "*strg5*" . data:n1 :sourceReport data:Rprt_x. …

Ein weiteres Beispiel für eine Reifikation ist die Modellierung der Kontaktdaten des Datenanbieters. Die Ressource ist als Individuum zu modellieren, da ihr später der Kontaktname sowie die E-Mail-Adresse zugeordnet werden müssen. Zunächst wird sie als Instanz der Klasse :Contact deklariert. Da sich auch Kontaktdaten ändern und somit doppeln können, wird daraufhin erneut ein Statement definiert, dem als Subjekt, Prädikat und Objekt die Informationen der zu treffenden Aussage zugeordnet werden und das als seinen Quellreport den Report Rprt_x angibt.

[175] Wie im nachfolgenden Kapitel über die URI-Vergabe thematisiert, ist der Organisationsname Teil des für den URI verwendeten Identifikators. Entsprechend würde bei einer Namensänderung eine neue Ressource für die Organisation des Datenanbieters angelegt werden. Allerdings ist eine Änderung des Organisationsname nicht gleichbedeutend mit einer Änderung der Organisation selbst, weshalb zwei getrennte Ressourcen für eine Organisation nicht sinnvoll sind. Die Lösung, ebenfalls im URI-Kapitel beschrieben, ist die Gleichsetzung der Ressourcen über owl:sameAs. Dadurch werden allerdings alle Eigenschaften der einen Ressource auch zu Eigenschaften der anderen - was zu dem Problem der zwei Labels an einer Ressource führt. Dasselbe Problem ist auch bei dem danach umgesetzten Organisationsnamen selbst vorhanden, weshalb dieser über eine Reifikation modelliert wird, was im Falle des Labels nicht möglich war.

<Contact>	data:Cntc_x a :Contact . data:n1 a rdf:statement . data:n1 rdf:subject data:Vndr_x . data:n1 rdf:predicate :hasContact. data:n1 rdf:object Cntc_x . data:n1 :sourceReport data:Rprt_x.

Darauf folgen schließlich noch die erwähnten Zuordnungen von Kontaktname und E-Mail, aufgrund auch derer Veränderlichkeit erneut als Reifikation.

In den Bereichen des Reports, die nicht wie die ressourcen- und organisationsspezifischen Daten über mehrere Reports auftreten, werden keine Reifikationen genutzt. Das durch das gewählte Beispiel entstandene Bild, dass Aussagen überwiegend mittels dieses Konstrukts getroffen würden, täuscht folglich. Der Großteil der Aussagen in einer RDF-Report-Datei erfolgt durch direkte Tripel, wie sehr gut an den im elektronischen Anhang D enthaltenen Testdateien zu sehen ist.

Die gesamte angelegte RDF/XML-Konkordanz mit einer Auflistung aller Elemente und deren Umsetzungen sowie Anmerkungen zu kritischen Stellen, ist in Anhang C verzeichnet.

5.4 URI-Vergabe

Alle bei der Überführung von XML nach RDF zu erstellenden Ressourcen, die keine Literale sind, müssen über eindeutige URIs für ihre Identifizierung verfügen. Der globale Teil des URIs ist über den Namensraum data bereits abgedeckt, indem folgende Platzhalter-Adresse den ersten Teil aller URIs darstellt:

```
http://www.platzhalter.de/erm/data#
```

Es bleibt also ein lokal eindeutiger Identifikator zu bestimmen. Die erste Forderung ist eben die Eindeutigkeit dieses Identifikators. Innerhalb des COUNTER-Namensraumes darf es für zwei unterschiedliche Ressourcen keine zwei gleichen Identifikatoren geben, da sonst beide Ressourcen den gleichen URI besäßen und somit als identisch angesehen würden.

Die zweite Forderung ist die Rekonstruierbarkeit des Identifikators. Neben nur einmal auftretenden, einzigartigen Daten wie etwa der Zählinstanz und dem Report gibt es auch die über mehrere Reports hinweg verzeichneten Daten, an denen sich zwar Details ändern, die aber insgesamt den gleichen Gegenstand beschreiben. Dieser eine Gegenstand soll möglichst auch in nur einer Ressource modelliert werden. Daraus ergibt sich, dass bei solchen reportübergreifenden Daten der Identifikator aus einem identifizierenden Merkmal der Daten selbst erstellt werden sollte. Dann erhalten alle aus späteren Daten modellierten Informationen denselben URI und werden automatisch als eine einzige RDF-Ressource modelliert.

Bei der Findung solcher Merkmale treten weitere Probleme auf. Nicht alle Elemente des XML-Schemas sind verpflichtend anzugeben. Stellt eine Information also theoretisch ein ideales identifizierendes Merkmal dar, kann sie in vielen Reports nicht enthalten sein. Für diesen Fall müssen Ausweichlösungen überlegt werden. Ein Beispiel ist die Identifikation von Instanzen der Klasse `:ReportItem` über den DOI. Dieser ist an sich bereits ein eindeutig und persistenter Identifikator für Medienwerke unterschiedlichster Art und somit hervorragend als Identifikator geeignet. Doch die Angabe eines DOIs zu einem Reportelement ist, wie auch die Angabe jeglicher anderer Item Identifier, im COUNTER-Schema nicht verpflichtend. Allein den DOI als Identifikator zu wählen, kann also bei fehlender Angabe eines DOIs im Report dazu führen, dass kein Identifikator gebildet werden kann. Daher wird alternativ auf die gleichfalls eindeutige ISSN bzw. ISBN des elektronischen Mediums zurückgegriffen. Im unwahrscheinlichen, aber formal dennoch möglichen Fall, dass auch diese nicht vorhanden ist, werden Titel und Verleger der Ressource für die Bildung des URI herangezogen. Diese Lösung hat den Nachteil, dass ein Medium zwar in einem Report keinen DOI haben kann, im nächsten aber schon - daraufhin wird es fälschlicherweise doch in zwei verschiedenen Ressourcen modelliert. Der Umgang mit solchen Fehlern wird weiter unten in diesem Kapitel thematisiert.

Nicht nur die fehlende Verpflichtung einer Angabe von Daten kann problematisch sein, sondern auch das schlichte Fehlen eines einzelnen, identifizierenden Merkmals. Hier müssen mehrere Merkmale miteinander kombiniert werden, bis von der Eindeutigkeit des gebildeten Identifikators ausgegangen werden kann. Dies ist beispielsweise bei Ressourcen der Fall, die Instanzen der Klasse `:Vendor` sind. Die Organisations-ID ist zwar verpflichtend anzugeben, allerdings wird sie vom Datenanbieter selbst festgelegt. Dies führt dazu, dass die ID zwar im System des einen Datenanbieters eindeutig sein sollte, sich aber trotzdem verschiedene Datenanbieter dieselbe Organisations-ID geben können. Zwei verschiedene Datenanbieter in einer Ressource zu modellieren, darf aber nicht geschehen. Also wird die Organisation-ID noch um den Organisationsnamen ergänzt, sofern diese

freiwillige Information angegeben wird. Die Übereinstimmung von selbst vergebener ID und Namen bei verschiedenen Datenanbietern ist unwahrscheinlich genug, um diese Merkmalskombination als Identifikator für den URI zu nutzen.

An dem Beispiel des Datenanbieters lässt sich gut ein weiteres Problem aufzeigen: Die geforderte Dauerhaftigkeit des Merkmals. Ändert sich der Organisationsname des Datenanbieters, wird eine neue Ressource angelegt, obwohl es sich noch immer um die gleiche Organisation handelt. Im konkreten Beispielfall ist davon auszugehen, dass sich Organisationsnamen relativ selten ändern, weshalb eine manuelle Fehlerkorrektur, wie weiter unten beschrieben, praktikabel ist. Generell sollte aber bei der Auswahl des identifizierenden Merkmals darauf geachtet werden, dass dieses eine möglichst hohe Konstanz erwarten lässt.

Einfach ist die Aufgabe der Findung eines Identifikators bei jenen bereits oben erwähnten Daten, welche nicht über mehrere Reports hinweg auftreten. Dazu gehören der Report selbst sowie die Zähleinheit. Hier müssen nicht bei einem nächsten Report Informationen als bereits vorhanden erkannt werden, denn diese Elemente sind einmalig. Jede Zähleinheit tritt nur einmal auf, jeder Report tritt nur einmal auf. Entsprechend ist es möglich, bei diesen Ressourcen einen aus einer laufenden Nummer bestehenden Identifikator für den URI zu nutzen. Dieser erfüllt alle Anforderungen.

Um die Identifikatoren verschiedener Ressourcenarten voneinander zu trennen, wie es etwa bei den laufenden Nummern von Report und Zähleinheit nötig ist, und um sie bereits auf den ersten Blick leichter einordnen zu können, steht am Anfang jedes Identifikators eine aus vier Buchstaben bestehende Abkürzung für dessen Bedeutung, abgegrenzt von den restlichen Zeichen durch einen Unterstrich. Diese Abkürzung soll sich beispielsweise, sofern vorhanden, aus der Bezeichnung der Klasse ergeben, zu der eine Ressource gehört. Die jeweils festgelegten Abkürzungen sind in Anhang C zur URI-Vergabe aufgeführt.

Einige Ressourcen stellen Daten dar, die bereits für sich Uniform Resource Identifier sind. Dies ist zum Beispiel bei der E-Mail-Adresse von bei Organisationen angegebenen Kontakten sowie Organisations-Websites und -Logos der Fall. Die E-Mail-Adresse ist ein URI des *mailto*-Schemas,[176] für Website und Logo werden URLs angegeben. Auch die Identifikatoren für Medien, der DOI, die ISSN sowie die ISBN, lassen sich dem URI-Schema gemäß darstellen. Ein DOI kann durch das Ergänzen

[176] Vgl. The 'mailto' URI Scheme, RFC 6068 [Elektronische Ressource], S. 8.

um das Präfix `http://doi.org/` als ein URL gemäß RFC 3986 ausgedrückt werden.[177] Eventuell für URIs nicht gültige Zeichen müssen kodiert werden, dazu unten mehr. ISBN und ISSN können als *Uniform Resource Name* (kurz URN) dargestellt werden. Die Darstellung ist einfach, da keine nicht gültigen Zeichen kodiert werden müssen, und sieht wie folgendes Beispiel aus:

```
urn:ISBN:978-0-06-232609-6
```

Die Bindestriche können beibehalten werden, sollten aber bei Vergleichen zweier URNs entfernt werden, ebenso sollte der teils anstelle der Prüfziffer genutzte Buchstabe X dafür groß geschrieben werden.[178] . Bei der ISSN verhält sich die Darstellung analog.[179] Für diese genannten Ressourcen werden also keine neuen URIs im `data`-Namespace vergeben, sondern die bereits vorgesehenen URIs genutzt.

Nachdem die zu verwendenden Daten und der Aufbau des URIs bestimmt sind, ist es nötig, die Zeichen in eine für URIs gültige Form zu bringen. Für die Definition von URIs werden die druckbaren Zeichen des ASCII-Zeichensatzes verwendet, von denen allerdings die in Tabelle 3 aufgeführten für strukturelle Kennzeichnungen reserviert sind.[180] Als unsicher gelten außerdem Leer- und Steuerzeichen, Zeichen mit unterschiedlichen nationalen Varianten sowie alle 8bit-Zeichen nach DEL (7F hex) des *ISO Latin-1*-Sets.[181]

:	/	?	#	[	]	/	@		
!	$	&	,	(	)	*	+	;	=

Tab. 3 Im URI-Schema reservierte ASCII-Zeichen.

Diese reservierten und unsicheren Zeichen sollen, um Probleme zu vermeiden, bei der Vergabe von URIs für Ressourcen in kodierter Form genutzt werden. Diese Funktion ist in OntoWiki bereits implementiert, wo entsprechend des RFC 3986 eine *Prozent-Kodierung*[182] (auch *URL-Endcoding*, *URL-Kodierung* oder engl. *Percent-Encoding*) auf Basis von UTF-8 durchgeführt wird. Dabei werden reservierte bzw. unsichere Zeichen in ihren Hexadezimalcode aufgelöst, wobei im Sinne der Datenkonsistenz die Verwendung von Großbuchstaben für die Zeichen A bis F empfohlen wird, und hinter ein Prozentzeichen gestellt.[183] Zwei Beispiele sollen das System verdeutlichen:

```
!     ⇨     %21
À     ⇨     %c3%80
```

[177] Vgl. DOI Handbook, Numbering, URI presentation [Elektronische Ressource].
[178] Vgl. Using International Standard Book Numbers as Uniform Resource Names, RFC 3187 [Elektronische Ressource], S. 3 f.
[179] Vgl. Using Existing Bibliographic Identifiers as Uniform Resource Names, RFC 2288 [Elektronische Ressource], S. 5.
[180] Vgl. Uniform Resource Identifier (URI): Generic Syntax, RFC 3986 [Elektronische Ressource], S. 13.
[181] Vgl. ebd., S. 12.
[182] Eine übersichtliche Darstellung findet sich unter http://www.backbone.se/urlencodingUTF8.htm.
[183] Vgl. Uniform Resource Identifier (URI): Generic Syntax, RFC 3986 [Elektronische Ressource], S. 12.

Aus diesen Überlegungen folgen für die Identifikatorvergabe von Ressourcen der genannten Beispielklassen die unteren Regelungen. Für die anderen Klassen des Schemas wurden, unter Berücksichtigung analoger Schwierigkeiten, ebenfalls Regeln aufgestellt, die in Anhang C aufgeführt sind.

Reportelement: `RpIt_ doi`*[DOI]*

alt. `issn`*[OnlineISSN]* bzw. `isbn`*[OnlineISBN]*

alt. *[ItemName]_[Vendor]*

```
Bsp.: ex:RpIt_doi10%2e1021%2fachre4
      ex:RpIt_issn1234-5678
      ex:RpIt_CAB%20Abstracts_Ovid%20Technologies%20Inc%2e
```

Report: `Rprt_` *[# laufende Nummer]*

```
Bsp.: ex:Rprt_42
      ex:Rprt_23
```

E-Mail: `mailto:`*[Email]*

```
Bsp.  <mailto: service@annualreviews.org>
      <mailto: support@ovid.org>
```

Sollte, wie bereits oben in Beispielen angedeutet, trotz dieser genannten Überlegungen zur sinnvollen Vergabe von URIs der Fall eintreten, dass ein einziger realer Gegenstand durch zwei unterschiedliche Ressourcen beschrieben wird, gibt es die Möglichkeit, diese beiden Ressourcen miteinander zu verbinden. OWL trifft keine *Unique Name Assumption,* geht also nicht davon aus, dass zwei unterschiedliche URIs auch tatsächlich zwei unterschiedliche Gegenstände beschreiben. Zur Lösung dieses Problems bietet OWL mit der Funktion `owl:sameAs` eine semantisch sehr starke Möglichkeit, zwei Individuen miteinander gleichzusetzen. Hierbei wird ausgesagt, dass ihre beiden URI-Referenzen auf den gleichen Gegenstand verweisen, sie also die gleiche Identität haben.[184] Diese Verbindung ist sowohl transitiv als auch symmetrisch und sollte daher nur sehr vorsichtig gebraucht werden: Sind zwei Ressourcen über `owl:sameAs` verknüpft, wird jede Eigenschaft der einen Ressource auch als Eigenschaft der anderen angesehen. Jede Veränderung an einer Ressource wirkt sich auch auf die andere aus. Nur wenn zwei Individuen wirklich absolut identisch sind, sollte auf `owl:sameAs` zurückgegriffen werden.[185] In der beschriebenen Problematik ist dies tatsächlich der Fall, da im optimalen Fall gar keine zweite, sondern nur eine einzige Ressource modelliert worden wäre.

[184] Vgl. OWL Web Ontology Language Guide, W3C Recommendation [Elektronische Ressource].
[185] Vgl. Halpin: When owl:sameAs isn't the same [Elektronische Ressource], S. 3 f.

6 Test des Vokabulars

6.1 Planung des Tests

Bevor das erstellte Vokabular endgültig im ERMS implementiert werden kann, ist es notwendig, seine Funktionalität zu testen. Dabei ist die erfolgreiche Umsetzung der zu Beginn der Modellierung an das Vokabular gestellten Anforderungen zu überprüfen. Weiterhin sollten jene Teile des Vokabulars, die bei der Modellierung Schwierigkeiten verursacht haben, mit besonderer Gründlichkeit betrachtet werden.

An das Vokabular wurden zu Beginn der Konzeption acht Hauptanforderungen gestellt:

1. Verhaltensanforderungen
   - *1.1 Vollständige Abbildung der XML-COUNTER-Reports ohne Informationsverlust.*
   - *1.2 Sinnvolle Navigation zwischen Reportbestandteilen.*
   - *1.3 Ständige Zuordnung Daten/Report möglich.*
2. Qualitätsanforderungen
   - *2.1 Deutsche und englische Erläuterungen.*
3. Problembereichsanforderungen
   - *3.1 Linked Data auf Basis von RDF.*
   - *3.2 Wiedergabe möglichst vieler Informationen aus dem XML-Schema.*
   - *3.3 Berücksichtigung von Besonderheiten von OntoWiki.*
   - *3.4 Ressource als Ausgangspunkt.*

Die unter den Punkten 2 und 3 aufgeführten Anforderungen können auch bereits ohne einen expliziten praktischen Test auf ihre Umsetzung überprüft werden. Jede im Vokabular modellierte Ressource besitzt deutsch- und englischsprachige Label, Klassen und Eigenschaften zudem auch Kommentare. Außerdem ist in der XML-RDF-Konkordanz festgelegt, dass bei der Erstellung von RDF-Ressourcen aus XML-Reports auch neu zu modellierenden Ressource entsprechende Label zugeordnet werden. Damit ist Anforderung 2.1 vollständig umgesetzt.

Da das erstellte Vokabular auf RDF basierend mit Hilfe von RDFS und OWL erstellt wurde, ist auch Anforderung 3.1 umgesetzt. Anforderung 3.2 wurde den Möglichkeiten entsprechend ebenfalls versucht umzusetzen, wobei durch den Linked-Data-Aufbau keine tatsächlichen Einschränkungen vorgenommen werden konnten: Die auf Datentypen bezogenen Einschränkungen werden durch Datatype Properties repräsentiert, die vorgeschriebene Kardinalität wird nur in Form einer Festlegung der minimalen Kardinalität wiedergegeben. Die Festlegungen von möglichen Werten für XML-Elemente werden als geschlossene OWL-Klassen in das Vokabular überführt. Anforderung 3.2 konnte also zumindest anteilig ebenfalls umgesetzt werden.

Der Aufbau des Vokabulars ist um die Reportelemente, also die Repräsentationen der bibliothekarischen Ressourcen, herum ausgerichtet. Der Anschluss an die anderen Module des ERMS erfolgt ebenfalls über die Reportelemente. Der Forderung nach einer Ressourcenzentrierung ist damit ebenfalls Folge geleistet.

Auf Besonderheiten von OntoWiki musste insofern Rücksicht genommen werden, dass bei der Konvertierung der Reportdaten in RDF auf Blanknodes, außer indirekt in Folge der `owl:oneOf`-Klassendefinitionen, verzichtet wird. Ob die `owl:oneOf`-Modellierung Schwierigkeiten verursacht, muss im Test überprüft werden, ist aber zu erwarten. Zunächst wird Anforderung 3.3, die Beachtung der Besonderheiten von OntoWiki, dennoch als umgesetzt angesehen. Ob noch weitere Änderungen erfolgen müssen, um eine reibungslose Integration in OntoWiki zu gewährleisten, wird sich durch den Test zeigen. In diesem Fall bliebe zu entscheiden, ob die Änderungen am Vokabular vorzunehmen sind oder ob OntoWiki in seiner Funktionalität zu ergänzen ist.

Ein Systemtest der Einheit *Nutzungsstatistiken* im vollständigen ERMS kann aufgrund des Arbeitsstandes am Gesamtprojekt noch nicht realisiert werden. Stattdessen soll das Vokabular gemeinsam mit einem Mitarbeiter der UB Leipzig in eine weitestgehend unangepasste OntoWiki-Installation eingeladen und anschließend eine Wissensbasis aus RDF-Reportdaten erstellt werden. Da der geplante XML/RDF-Konverter erst nach Abschluss dieser Masterarbeit realisiert werden kann, ist es nicht möglich, einen Test mit real konvertierten RDF-Reportdaten durchzuführen. Stattdessen werden die XML-Daten manuell in eine Turtle-Serialisierung überführt.

Der Test selbst besteht größtenteils aus der manuellen Kontrolle der Anzeige und der Verknüpfungen der Daten. Dazu sollen die verschiedenen Testdaten stichprobenartig einzeln aufgerufen werden und auf die korrekte Darstellung der richtigen, mit dieser Ressource korrespondierenden Daten

überprüft werden. Es soll, ebenfalls stichprobenartig, überprüft werden, ob Ressourcen fälschlicherweise doppelte Daten enthalten oder doppelte Ressourcen für einzelne Gegenstände erstellt wurden, was auf eine inkorrekte URI-Vergabe schließen ließe. Weiterhin soll geprüft werden, ob Daten aller Klassen über Verbindungen mit anderen Daten aufrufbar sind und ob die modellierte Verknüpfungsstruktur auch in der Bedienung sinnvoll angelegt ist. Neben dem Abruf von Daten über die Verfolgung der Graphenverknüpfungen sollen auch einige SPARQL-Anfragen an das System gestellt werden.

Diese Art des Tests ist sehr subjektiv angelegt. Außer bei den SPARQL-Anfragen ist es nicht möglich, vor dem Test zu erwartende Ausgabewerte zu bestimmen und diese anschließend zu kontrollieren. Diese Subjektivität ließe sich zu einem gewissen Grad durch eine höhere Formalisierung des geplanten Testvorganges ausgleichen. Da aber sowohl die Oberfläche als auch die Funktionalität der für den Test nutzbaren OntoWiki-Installation noch nicht der endgültigen Form entsprechen, wurde ein formal korrekter Test in dieser Phase noch für unnötig befunden. Dieser sollte im Zuge eines Tests des Gesamtsystems nachgeholt werden.

6.2 Erstellung von Testdatensätzen

Für den manuellen Test hat die UB Leipzig einige von verschiedenen Anbietern abgerufene XML-Dateien unverändert zur Verfügung gestellt. Bis auf eine Ausnahme beinhalten alle Dateien mehrere unterschiedliche Reports. Der hohe Aufwand, diese umfangreichen Dateien vollständig manuell in RDF zu konvertieren, erscheint für den geplanten Test nicht notwendig, wenn unter Berücksichtigung der Testziele bestimmte Teile aus den Testdateien zur Konvertierung ausgewählt werden.

Zur Kontrolle der Vollständigkeit der Infomationsübernahme ist es wichtig, möglichst unterschiedliche Daten zu konvertieren. Das Testen einer sinnvollen Navigation erfordert daneben auch ähnliche Daten, um die Unterscheidbarkeit gleichartig strukturierter Informationen zu kontrollieren. Für die Überprüfung der korrekten Zuordnung von Daten und Reports sowie der eindeutigen Bildung von URIs ist es ebenfalls notwendig, ähnliche Daten zu erstellen.

Daher wurde festgelegt, dass insgesamt zehn Reports unterschiedlicher Art konvertiert werden sollen. Darunter befinden sich sechs Journal- und zwei Book-Reports jeweils verschiedener Varianten, ein Titel Report, ein Plattform- sowie ein Datenbankreport. Das mögliche Spektrum an Reportarten ist also recht gut abgedeckt, der Schwerpunkt auf den Journal-Reports entspricht der momentanen ERMS-Realisierung an der UB Leipzig, die sich auf Zeitschriften konzentriert. Aus den meisten Reports konnten jeweils zwei Reportelemente ausgewählt werden, um mehrere Elemente mit einem Report verbinden zu können. Einige der Journal-Reports berichten über die gleichen Reportelemente. Die Reports stammen außerdem von nur drei verschiedenen Datenanbietern, sodass auch in diesem Bereich sowohl gleichartige, als auch unterschiedliche Daten produziert werden.

Bei der Erstellung der Daten wurden die Regelungen der XML-RDF-Konkordanz befolgt. Daten, die dem manuellen Ersteller als identisch auffallen konnten, wurden trotzdem, wie auch später durch das Konvertierungstool, doppelt modelliert. Dadurch kann überprüft werden, ob diese Daten vom Tripelstore tatsächlich anhand ihrer theoretisch identischen URIs gleichgesetzt werden. Allerdings wurden der Schreibarbeit wegen im Gegensatz zu der in der Konkordanztabelle angegebenen Darstellung mehr Turtle-Verkürzungen genutzt und die Reihenfolge einiger Elemente in der Code-Abfolge verändert. Diese Änderungen haben allerdings keine Auswirkung auf die Tripel, die aus der Turtle-Datei gewonnen werden, wodurch sich für das Testergebnis daraus keine Einschränkungen ergeben.

Insgesamt wurden aus drei XML-Dateien 17 Reportelemente mit zusammen 374 Zählinstanzen modelliert, die in drei Turtle-Dateien gespeichert wurden. Diese Testdateien sind in Anhang D auf der beiliegenden CD-ROM gespeichert.

Reportart	**Datenanbieter**	**Report-elemente**
BR2	American Chemical Society	2
BR3	American Chemical Society	2
DB1	Ovid Technologies Inc.	2
JR1	American Chemical Society	2
JR1GOA	Informa UK Ltd.	1
JR2	American Chemical Society	2
JR3	American Chemical Society	2
JR5	American Chemical Society	1
PR1	American Chemical Society	2
TR1	American Chemical Society	1

Tab. 4 *Zur Modellierung von Testdatensätzen verwendete Informationen.*

6.3 Testergebnisse

Die Vollständigkeit der Reportabbildung konnte bereits während der Erstellung der eigentlichen Testdaten überprüft werden. Die manuelle Erstellung der RDF-Dateien war hier sogar von Vorteil, da jeder Report Element für Element komplett durchgegangen wurde. Dabei wurde jede Information des Reports übertragen. Die einzige Ausnahme bildeten die ausgeschriebenen Titel der Reports. Diese werden nicht aus dem XML-Dokument gewonnen, sondern können aufgrund ihrer Standardisierung bereits im Vokabular dem jeweiligen Kurztitel zugeordnet werden. Dadurch geht bei schemakonformen Reports keine Information verloren. Im DB1-Report von Ovid war statt des regulären Titels *Database Report 1* allerdings der ausführlichere Titel *DataBase Report 1 (R4): Total Searches, Result Clicks and Record Views by Month and Database* angegeben, der auf die von COUNTER genutzte Beschreibung zurückgeht. Dieser Titel konnte entsprechend nicht direkt übernommen werden, er ist allerdings in Form eines bereits im Vokabular integrierten Kommentars der Instanz `count:DB1` trotzdem Teil der RDF-Darstellung. Von dieser Ausnahme abgesehen, verlief die manuelle Konvertierung problemlos und verlustfrei.

Die erstellten Testdateien sowie das Vokabular wurden anschließend als Wissensbasis in OntoWiki eingeladen, was ebenfalls problemlos ablief. Die Anzeige ersetzte die URIs der Ressourcen korrekt durch Labels, sofern solche vorhanden waren, was zu einem gut lesbaren Gesamtbild führte. Die Sprache der Labels sowie der Kommentare konnte zwischen deutschen und englischen Bezeichnungen umgeschaltet werden. Es sind keine doppelten Ressourcen aufgefallen, die auf eine fehlerhafte URI-Vergabe hätten schließen lassen können. Ebenso wurden keine Ressourcen gefunden, die fälschlicherweise Informationen zweier verschiedener Einheiten enthielten, also die Folge eines nicht eindeutigen URIs gewesen wären.

Die RDFS-Klassen wurden allesamt als solche erkannt und entsprechend dargestellt, ihre Instanzen waren ihnen jeweils zugeordnet. Die mit `owl:Cardinality`-Einschränkungen versehenen owl-Klassen waren ebenfalls korrekt eingebettet. Wie erwartet gab es allerdings Probleme bei den geschlossenen OWL-Klassen, deren Instanzen mittels `owl:oneOf` aufgezählt werden. Die in dieser Konstruktion genutzten Blanknodes konnten wie erwartet von OntoWiki nicht richtig verarbeitet werden, die Klassen fehlten in der Klassenübersicht. Bei Aufruf der Detailansicht für Instanzen dieser Klassen, wurde allerdings der `rdf:type` korrekt angezeigt, das Individuum wurde korrekt als Instanz der Klasse erkannt. Der Fehler beschränkte sich offenbar lediglich auf die Anzeige in der Klassenübersicht.

Auch die Anzeige der Reifikationen bereitete leichte Schwierigkeiten. Beispielsweise wurde in der Detailansicht der Ressource `data:Vndr_Ovid_Ovid%20Technologies%20Inc%2e`, die den Datenanbieter Ovid darstellt, nicht angezeigt, dass dieser Ressource laut mehreren Reports ein Organisationsname zugeordnet wird. Es war lediglich in der Anzeige der mit dieser Ressource verknüpften weiteren Ressourcen das Statement sichtbar, welches diese Aussage trifft. Bei Aufruf dieses Statements wird die Aussage korrekt angezeigt. Auch mehrere testweise formulierte SPARQL-Queries lieferten korrekte Ergebnisse. Es handelte sich hierbei also um keinen tatsächlichen Fehler, sondern lediglich um eine nicht nutzerfreundliche Darstellung der Reifikation. Die Anforderung der vollständigen Übertragung der XML-Informationen wird also erfüllt.

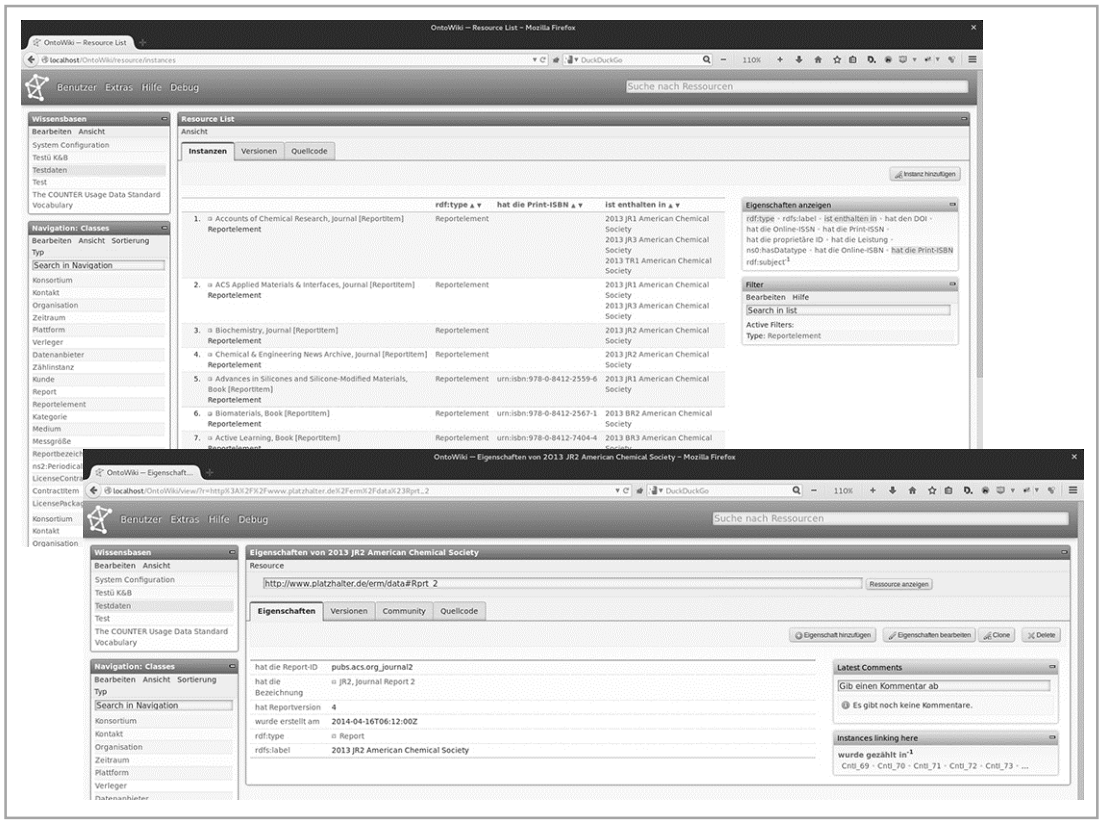


Abb. 20 *Screenshots, OntoWiki mit eingeladenen Testdateien. Vorn: Detailansicht eines Reports, hinten: Übersicht aller Reportelemente.*

Alle anderen testweise genutzten Verknüpfungen und Detailanzeigen funktionierten problemlos. Ausgehend von einem Reportelement ließ sich sowohl dessen Report als auch die Zähleinheit aufrufen, diese wiederum zeigte korrekt alle ihr zugeordneten Daten an. Sie verwies ihrerseits auf den Kunden, für dessen Lizenzen die Nutzung gemessen wurde, sowie auf den sie beinhaltenden Report. Von diesem Report aus waren alle Reportelemente und Zähleinheiten abrufbar, die aus seinen Daten erstellt wurden. Einem anderen Ansatzpunkt folgend konnten zu einem bestimmten Zeitraum alle in diesem gemessenen Zählinstanzen angezeigt werden. Über diese Zähleinheiten war

die Anzeige der entsprechenden Reportelemente möglich. Analog wurden alle weiteren vorgesehenen Verknüpfungen mit positivem Ergebnis überprüft. Die genannten Verknüpfungen boten dabei allerdings nur einen Einstieg in die Möglichkeiten des Informationsabrufs. Über den integrierten SPARQL-Endpunkt konnten alle gestellten, auch komplexeren Anfragen korrekt beantwortet werden.

Insgesamt können folglich auch die Forderungen nach einer guten Navigationsmöglichkeit sowie nach einer ständigen Zuordnungsmöglichkeit von Daten und Reports als erfüllt angesehen werden. Bei beiden Umsetzungen existieren zwar unter Nutzung von OntoWiki momentan noch Schwierigkeiten, allerdings sind sowohl Navigation als auch Zuordnung durch entsprechende SPRQL-Anfragen sehr gut möglich. Darüber hinaus kann OntoWiki entsprechend angepasst werden, um eine bestmögliche Bedienbarkeit zu erreichen.

7 Fazit und Ausblick

Das primäre Ziel der vorliegenden Arbeit war die dokumentierte Erstellung eines RDF-Vokabulars, welches die Grundlage bieten sollte, die gesamten in COUNTER-konformen Reports enthaltenen Informationen zu erfassen und miteinander zu verknüpfen, sodass eine RDF-basierte Verarbeitung von Nutzungsstatistiken unter anderem im ERMS der UB Leipzig ermöglicht wird.

Zum Erreichen dieses Ziels wurde zunächst auf Basis von RDFS ein Vokabular modelliert, das aus an das COUNTER-XML-Schema angelehnten Klassen und Eigenschaften besteht. Mit dessen Hilfe lassen sich bereits alle Informationen, die in COUNTER-konformen Reports zusammengefasst werden können, als RDF-Ressourcen verknüpft darstellen. Um auch für anders akzentuierte Nachnutzungen offen zu sein und die Daten semantisch weiter anzureichern, wurde anschließend mit Hilfe von OWL ein Großteil der restlichen im XML-Schema vorhandenen Informationen sinngemäß angepasst und in das Vokabular übertragen. Nach weiterführenden Überlegungen zur Umsetzung der konkreten Reportdaten in Turtle-Code und zur Identifizierung der zu erstellenden Ressourcen, war es möglich, reale XML-Reports manuell vollständig in RDF-Daten umzuwandeln und diese erfolgreich zu testen. Die in dieser Arbeit geleistete Modellierung kann daher, vorerst noch losgelöst von der tatsächlichen praktischen Implementierung im Bibliotheksalltag, als erfolgreich und die Ziele der Arbeit als erfüllt angesehen werden.

Für die konkrete Anwendung wurden alle genannten Vorgaben, wie etwa die vorerst noch nicht durchzuführende Nachnutzung anderer Vokabulare sowie die weitgehende Vermeidung von Blanknodes, berücksichtigt. Auch prinzipielle Überlegungen, die der Verwendung im auf längerfristige Speicherung angelegten ERM-System geschuldet sind, sind in den Entwurf eingeflossen, so etwa die Möglichkeit der historischen Zuordnung von Daten durch die Verwendung von reifizierten Aussagen. Mit Hilfe der XML-RDF-Konkordanz sowie den Überlegungen zur URI-Vergabe wird die Erstellung des Konverters sowie die Implementierung in das Gesamt-ERMS vereinfacht. Gleichzeitig wurde die Modellierung nicht auf den Einsatz in der UB Leipzig reduziert, sondern das gesamte COUNTER-XML-Schema in die Linked-Data-Welt zu übertragen versucht. Dabei konnten zwar die

im XML vorhandenen Einschränkungen nicht direkt übernommen werden, diese sind jedoch in analoge OWL-Konstrukte überführt worden. Insgesamt kann das Vokabular also auch für diverse andere Anwendungen rund um COUNTER-Nutzungsstatistiken genutzt werden.

Ob bei dem tatsächlichen praktischen Einsatz in der UB Leipzig noch Probleme auftreten werden, ist zum derzeitigen Entwicklungsstand des Gesamtsystems noch nicht erkennbar. Der kurze Test lässt allerdings zumindest grobe Fehler und Probleme eher unwahrscheinlich wirken. Die Fehler bei der Verarbeitung von Blanknodes in OntoWiki, die unter anderem bei den mittels `owl:oneOf` definierten Klassen zum Tragen kamen, sowie die Anzeigeschwierigkeiten bei reifizierten Aussagen konnten im Rahmen dieser Arbeit nicht gelöst werden. Auch die mögliche Doppelung von Labels bestimmter Ressourcen, die bei der Erläuterung der XML-RDF-Übertragung als kritisch eingestuft wurde, konnte nicht vermieden werden. Dies sind somit Aufgabe für Folgearbeiten im Projekt.

Ebenso muss das Tool zur XML/RDF-Konvertierung noch geschrieben werden, wofür mit der erstellten Konkordanz der Grundstein gelegt wurde. Für die komfortable Datenabfrage in OntoWiki sollten einige feste SPARQL-Queries für Standardabfragen innerhalb des Gesamtsystems erstellt werden. Anwendenden Bibliotheksmitarbeitern könnte ansonsten insbesondere die durch reifizierte Aussagen umgesetzte Historie Schwierigkeiten bereiten. Das ganze Potential der verknüpften Reportdaten kommt dabei aber erst im Zusammenspiel mit den anderen Komplexen des ERMS zum Tragen, wenn beispielsweise direkt zu Lizenzverträgen bestimmter Jahre die Nutzungsdaten abrufbar sind und somit ein schneller Kosten-Nutzen-Vergleich möglich wird.

Neben den noch vor der Implementierung notwendigen Arbeiten wird das Vokabular kontinuierlich gepflegt werden müssen. Das Projekt COUNTER arbeitet fortwährend daran, seine Standards den sich rasch wandelnden Gegebenheiten auf dem Markt der elektronischen Ressourcen anzupassen. Das XML-Schema ist zwar auf eine dauerhafte Nutzung auch mit späteren Versionen des CoP ausgelegt, allerdings bildet sein Linked-Data-Äquivalent auch die eher veränderliche COUNTER Elements-Datei ab. Insofern müssten mögliche Änderungen insbesondere für die geschlossen definierten Klassen, längerfristig aber auch für andere Teile des Vokabulars berücksichtigt werden. Es existiert kein fester Zeitplan für das Erscheinen neuer COUNTER CoPs, aufgrund der bisherigen Zeiträume zwischen zwei Versionsveröffentlichungen lässt sich aber ungefähr mit einem Abstand von zwei bis drei Jahren rechnen.[186]

[186] Vgl. COUNTER FAQs [Elektronische Ressource].

Eine weitere noch zu erledigende Aufgabe wird das Analysieren von anderen Linked-Data-Vokabularen im Sinne einer Nachnutzung darstellen, um eine umfangreichere semantische Beschreibung der Daten zu erreichen und doppelte Modellierungen zu vermeiden. Analog zum restlichen System[187] sollten für die bibliographischen Daten die Ontologien der *Dublin Core Meta Data Initiative* (DCMI)[188] sowie die *Bibliographic OntologySpecification* (BIBO)[189] auf zu verwendende Elemente untersucht werden. Für Organisationen und Kontakte betreffende Daten sind demnach die *Academic Institution Internal Structure Ontology* (AIISO)[190] sowie *das Friend of a Friend Vocabulary* (FOAF)[191] auszuwerten. Nicht zuletzt sollten, sofern möglich, auch Teile des im Rahmen des AMSL-Projektes modellierten Vokabulars BIBRM genutzt werden. Im Zuge dessen sind auch die Regelungen zur URI-Vergabe an entsprechenden Stellen dem restlichen System anzupassen. So sollten etwa zur Identifikation und Verbindung genutzte Ressourcen wie die ISSN in allen Teilen des ERMS identisch aufgebaut werden, um eine optimale Kompatibilität zu gewährleisten.

Das AMSL-Projekt wird in den nächsten Monaten zum Abschluss kommen, der Einsatz des Systems wird entsprechend bald anlaufen. Dieser Einführungsprozess wird sich interessant gestalten, nicht nur auf Ebene der nun hausintern möglichen, gut organisierten Verwaltung elektronischer Ressourcen, sondern auch in Bezug auf die in diesem Bereich erfolgende Kooperation der sächsischen Hochschulbibliotheken. In diesem Kontext sei auf den in Kürze erscheinenden Artikel zum Projekt hingewiesen, der ausführlicher auf Ziele und Umsetzung des Gesamtprojektes eingeht.

Aufgrund der Möglichkeit der umfassenden Abbildung von COUNTER-Reports könnte das erstellte Vokabular eine gute Arbeitsgrundlage für die Einführung von RDF-basierten Reports darstellen. In diesem Sinne soll es als Idee an das Projekt COUNTER herangetragen werden und mag so dazu beitragen, standardisierte Nutzungsstatistiken zukünftig direkt im Linked-Data-Format anbietbar zu machen.

[187] Vgl. AMSL [Elektronische Ressource], S. 5.
[188] Siehe unter http://purl.org/dc/terms/ sowie http://purl.org/dc/elements/1.1/.
[189] Siehe unter http://bibliontology.com/.
[190] Siehe unter http://vocab.org/aiiso/schema.
[191] Siehe unter http://xmlns.com/foaf/0.1/.

Literatur- und Quellenverzeichnis

About AMSL [Elektronische Ressource]. - Online-Ressource
Adresse: http://amsl.technology/about-amsl/
Gesehen: 18.06.2014

About COUNTER [Elektronische Ressource]. - Online Ressource. - Stand: März 2014
Adresse: http://projectcounter.org/about.html
Gesehen: 17.06.2014

Allemang, Dean:
Semantic web for the working ontologist : effective modeling in RDFS and OWL / Dean Allemang ; Jim Hendler. - 2nd ed. - Amsterdam [u.a.] : Elsevier, 2011. - XIII, 354 S. : graph. Darst.
ISBN 978-0-12-385965-5

AMSL [Elektronische Ressource] : managing electronic resources for libraries based on semantic web / Andreas Nareike ; Natanael Arndt ; Norman Radtke … - Stand: 01.07.2014. - 11 S. (0,1 MB)
Adresse: Bisher unveröffentlicht.

BIBFRAME - Bibliographic Framework Initiative [Elektronische Ressource] / Deutsche National-bibliothek. - Online-Ressource. - Stand: 23.04.2014
Adresse: http://www.dnb.de/bibframe
Gesehen: 29.06.2014

Bitte testen: Alpha-Release unserer Datenmanagementsoftware [Elektronische Ressource] / Jens Mittelbach. - Online-Ressource. - Stand: 30.05.2014
Adresse: http://dmp.slub-dresden.de/2014/05/bitte-testen-alpha-release-unserer-datenmanage-ment-software/
Gesehen: 06.06.2014

CamelCase [Elektronische Ressource]. - Online-Ressource. - Stand 12.04.2014. - (C2 Wiki)
Adresse: http://c2.com/cgi/wiki?CamelCase
Gesehen: 04.06.2014

The **COUNTER Code of Practice for e-Resources: Release 4** [Elektronische Ressource] / COUNTER Online metrics. - Online-Ressource. - April 2012. - 29 S. (3,2 MB)
Adresse: http://projectcounter.org/r4/COPR4.pdf
Gesehen: 17.06.2014

COUNTER FAQs [Elektronische Ressource]. - Online-Ressource. - Stand: März 2014
Adresse: http://projectcounter.org/faqs.html
Gesehen: 17.06.2014

COUNTER members [Elektronische Ressource]. - Online-Ressource. - Stand: Juni 2014
Adresse: http://projectcounter.org/members.html
Gesehen: 17.06.2014

COUNTER register of vendors [Elektronische Ressource]. - Online-Ressource. - Stand: Juni 2014
Adresse: http://projectcounter.org/compliantvendors.html
Gesehen: 17.06.2014

COUNTER Schema Data Element Values [Elektronische Ressource] : Code of Practice Release 4 / National Information Standards Organization. - Online-Ressource
Adresse: http://www.niso.org/workrooms/sushi/values/
Gesehen: 01.06.2014

COUNTER Usage factor [Elektronische Ressource]. - Online-Ressource. - Stand: April 2014
Adresse: http://projectcounter.org/usage_factor.html
Gesehen: 17.06.2014

Data Strategy and Linked Data [Elektronische Ressource] / OCLC - Online Computer Library Center. - Online-Ressource. - Stand: 29.05.2014
Adresse: http://www.oclc.org/data.en.html
Gesehen: 29.06.2014

Dietzold, Sebastian:
Koll[a]borative Wissensarbeit mit OntoWiki [Elektronische Ressource] / Sebastian Dietzold, Sören Auer, Thomas Riechert. - Online-Ressource. - Stand: 18.06.2006. - 8 S. (0,6 MB)
Adresse: http://www.informatik.uni-leipzig.de/~auer/publication/WissensarbeitMitOntoWiki.pdf
Gesehen: 29.06.2014

DOI Handbook, Numbering, URI presentation [Elektronische Ressource] / International DOI Foundation. - Online-Ressource. - Stand 13.02.2014
Adresse: http://www.doi.org/doi_handbook/2_Numbering.html#2.6.2
Gesehen: 25.06.2014

E-Books and ISBNs [Elektronische Ressource] : a position paper and action points from the International ISBN Agency. - Online-Ressource. - Stand: 26.02.2010 - 3 S. (35 KB)
Adresse: http://www.isbn.org/sites/default/files/images/isbn_agency_e-books_position_paper.pdf
Gesehen: 05.05.2014

Electronic Resource Management: Report of the DLF ERM Initiative [Elektronische Ressource] / Ivy Anderson ; Sharon E. Farb ; Adam Chandler … - Online-Ressource - Washington D.C. : Digital Library Federation, 2004. - 267 S. (1,5 MB)
Adresse: http://old.diglib.org/pubs/dlf102/ERMFINAL.pdf
Gesehen: 17.06.2014

Forward to the Report of the DLF ERM Initiative [Elektronische Ressource] / David Seaman. - Online-Ressource. - Washington, D.C. : Digital Library Federation, 2004
Adresse: old.diglib.org/pubs/dlf102/
Gesehen: 16.06.2014

Das **Graphenmodell von RDF** [Elektronische Ressource] : Seminar an der Deutschen Bibliothek. - Online-Ressource. - Stand: 16.06.2000 - 12 S. (0,1 MB)
Adresse: http://www.iwi-iuk.org/seminarNotes/1/jgrafexp.pdf
Gesehen: 27.05.2014

Halpin, Harry:
When owl:sameAs isn't the same [Elektronische Ressource] : an analysis of identity links on the semantic web / Harry Halpin ; Patrick J. Hayes. - Online Ressource. - Stand: 25.03.2010. - 5 S. (84 KB)
Adresse: http://events.linkeddata.org/ldow2010/papers/ldow2010_paper09.pdf
Gesehen: 24.06.2013

Knauber, Peter:
Anforderungsanalyse [Elektronische Ressource] : Vorlesung Software Engineering / Peter Knauber. - Online-Ressource. - Hochschule Mannheim, 2006. - 47 Folien (0,4 MB ; 0,7 MB)
1. Adresse: http://services.informatik.hs-mannheim.de/~knauber/BCSc-SE/06a-g.pdf
2. Adresse: http://services.informatik.hs-mannheim.de/~knauber/BCSc-SE/06b-g.pdf
Gesehen: 27.05.2014

Kowalak, Mario:
Electronic Resource Management (ERM) in Bibliotheken [Elektronische Ressource] : eine knappe Einführung ; Vortrag bei den 19. Gemeinsamen Bibliothekstagen für Niedersachsen und Sachsen-Anhalt / Mario Kowalak. - Online-Ressource. - Stand: 01.12.2011. - 29 Folien (1,6 MB)
Adresse: http://www.bibliotheksverband.de/fileadmin/user_upload/Landesverbaende/Niedersachsen/Vortrag_Kowalak.pdf
Gesehen: 13.06.2014

Library Linked Data Incubator Group [Elektronische Ressource]. - Online-Ressource. - Stand: 06.08.2013. - (W3C wiki)
Adresse: http://www.w3.org/2005/Incubator/lld/wiki/Main_Page
Gesehen: 29.06.2014

Librdf FAQ [Elektronische Ressource] / Dave Becket. - Online-Ressource. - Stand: 30.01.2014
Adresse: http://librdf.org/FAQS.html
Gesehen: 31.05.2014

Linked Data [Elektronische Ressource] : Design Issues / Tim Berners-Lee. - Online-Ressource. - Stand: 18.06.2009
Adresse: http://www.w3.org/DesignIssues/LinkedData.html
Gesehen: 30.05.2014

Linked Data glossary [Elektronische Ressource] / W3C Government Linked Data Working Group. - Online-Ressource. - Stand: 27. 06 2013
Adresse: http://www.w3.org/TR/2013/NOTE-ld-glossary-20130627/
Gesehen: 03.04.2014

Linked Data Service der Deutschen Nationalbibliothek [Elektronische Ressource]. - Online-Ressource. - Stand: 18.06.2014
Adresse: http://www.dnb.de/lds
Gesehen: 29.06.2014

The **'mailto' URI Scheme, RFC 6068** [Elektronische Ressource] / M. Duerst, L. Masinter, J. Zawinski - Online Ressource. Stand: Oktober 2010. - 17 S.
Adresse: http://tools.ietf.org/html/rfc6068
Gesehen: 24.06.2014

News zu AMSL [Elektronische Ressource]. - Online-Ressource
Adresse: http://amsl.technology/
Gesehen: 17.06.2014

Notepad++ [Elektronische Ressource] / Don Ho. - Online-Ressource
Adresse: http://notepad-plus-plus.org/
Gesehen: 01.06.2014

Notepad++ Contributors [Elektronische Ressource] / Don Ho. - Online-Ressource
Adresse: http://notepad-plus-plus.org/contributors/
Gesehen: 01.06.2014

Notepad++ Features [Elektronische Ressource] / Don Ho. - Online-Ressource
Adresse: http://notepad-plus-plus.org/features/
Gesehen: 01.06.2014

Notepad++ Wiki - User Defined Language Files [Elektronische Ressource] : SourceForge. - Online-Ressource. - Stand: 30.05.2014
Adresse: http://sourceforge.net/apps/mediawiki/notepad-plus/index.php?title=User_Defined_Language_Files
Gesehen: 01.06.2014

OntoWiki [Elektronische Ressource] : a tool providing support for agile, distributed knowledge engineering scenarios. - Online-Ressource
Adresse: http://aksw.org/Projects/OntoWiki.html
Gesehen: 29.06.2014

OWL Web Ontology Language Current Status [Elektronische Ressource]. - Online-Ressource. - Stand: 26. Juni 2014.
Adresse: http://www.w3.org/standards/techs/owl#w3c_all
Gesehen: 28.06.2014

OWL Web Ontology Language Guide, W3C Recommendation [Elektronische Ressource] / Michael K. Smith ; Chris Welty ; Deborah L. McGuinness. - Online-Ressource. - Stand: 10.02.2004
Adresse: http://www.w3.org/TR/owl-ref/
Gesehen: 12.06.2014

OWL Web Ontology Language Reference, W3C Recommendation [Elektronische Ressource] / Sean Bechhofer ; Frank van Harmelen ; Jim Hendler … - Online-Ressource. - Stand: 10.02.2004
Adresse: http://www.w3.org/TR/owl-ref/
Gesehen: 12.06.2014

OWL 2 Web Ontology Language Primer (Second Edition), W3C Recommendation [Elektronische Ressource] / Pascal Hitzler ; Markus Krötzsch ; Bijan Parsia … - Online-Ressource. - Stand: 11.12.2012
Adresse: http://www.w3.org/TR/2012/REC-owl2-primer-20121211/
Gesehen: 12.06.2014

Powers, Shelley:
Practical RDF : [solving problems with the resource description framework] / Shelley Powers. - 1. ed. - Beijing [u.a.] : O'Reilly, 2003. — XV, 331 S. : graph. Darst.
ISBN 0-596-00263-7

Projekt Datenmanagement und ERM [Elektronische Ressource] / Jens Mittelbach. - Online-Ressource. - Stand: 13.12.2013
Adresse: http://dmp.slub-dresden.de/ueber-das-projekt/
Gesehen: 18.06.2014

Radtke, Norman [Gespräch]. - Persönliches Gespräch mit Autorin in der Universitätsbibliothek Leipzig. - 27.06.2014

Raptor Download Win32 [Elektronische Ressource] / Dave Becket. - Online-Ressource. - Stand: 28.11.2009
Adresse: http://download.librdf.org/binaries/win32/README.txt
Gesehen: 31.05.2014

Raptor RDF Syntax Library [Elektronische Ressource] / Dave Becket. - Online-Ressource. - Stand: 05.05.2014
Adresse: http://librdf.org/raptor/
Gesehen: 31.05.2014

RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation [Elektronische Ressource] / Richard Cyganiak, David Wood, Markus Lanthaler. - Stand: 25.02.2014
Adresse: http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/
Gesehen: 21.06.2014

RDF 1.1 Schema, W3C Recommendation [Elektronische Ressource] / Dan Brickley, R.V. Guha. - Online-Ressource. - Stand: 25.02.2014
Adresse: http://www.w3.org/TR/rdf-schema/
Gesehen: 21.06.2014

RDF 1.1 Turtle, W3C Recommendation [Elektronische Ressource] / David Beckett ; Tim Berners-Lee ; Eric Prud'hommeaux … - Online-Ressource. - Stand: 25.02.2014
Adresse: http://www.w3.org/TR/2014/REC-turtle-20140225/
Gesehen: 05.05.2014

Redland RDF Libraries [Elektronische Ressource] / Dave Becket. - Online-Ressource. - Stand: 11.05.2014
Adresse: http://librdf.org/
Gesehen: 31.05.2014

Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation [Elektronische Ressource] / Ora Lassila, Ralph R. Swick. - Online-Ressource. - Stand 22.02.1999
Adresse: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/
Gesehen:25.06.2014

Schemas for the Standardized Usage Statistics Harvesting Initiative (SUSHI) [Elektronische Ressource] / National Information Standards Organization. - Online-Ressource
Adresse: http://www.niso.org/schemas/sushi/
Gesehen: 01.06.2014

Semantic Web : Wege zur vernetzten Wissensgesellschaft; mit ... 4 Tab. / Tassilo Pellegrini ... (Hrsg.). - Berlin [u.a.] : Springer, 2006. - XIV, 533 S. : Ill., graph. Darst. - (X.media.press)
ISBN 3-540-29324-8 - ISBN 978-3-540-29324-8

Semantische Wikis [Elektronische Ressource] / Sebastian Schaffert, Francois Bry, Joachim Baumeister … - Online-Ressource. - Stand: 06.11.2008. - 15 S. (1,1 MB)
Adresse: http://www.en.pms.ifi.lmu.de/publications/PMS-FB/PMS-FB-2009-1/PMS-FB-2009-1-paper.pdf
Gesehen: 29.06.2014

Stuckenschmidt, Heiner:
Ontologien : Konzepte, Technologien und Anwendunge / Heiner Stuckenschmidt. - 2. Aufl. - Berlin [u.a.] : Springer, 2011. - XIII, 273 S. : graph. Darst. - (Informatik im Fokus)
ISBN 978-3-642-05403-7

SUSHI Reports Registry [Elektronische Ressource] : COUNTER Code of Practice Release 4 / National Information Standards Organization. - Online-Ressource
Adresse: http://www.niso.org/workrooms/sushi/reports/
Gesehen: 01.06.2014

SUSHI/COUNTER Schemas [Elektronische Ressource] / National Information Standards Organization. - Online-Ressource.
Adresse: http://www.niso.org/schemas/sushi/
Gesehen: 18.06.2014

Uniform Resource Identifier (URI): Generic Syntax, RFC 3986 [Elektronische Ressource] / T. Berners-Lee, R. Fielding, L. Masinter. - Online-Ressource. - Stand: Januar 2005. - 61 S.
Adresse: http://tools.ietf.org/html/rfc3986
Gesehen: 24.06.2014

Unterdörfel, Lydia:
Entwicklung eines Electronic Resource Management Systems für Bibliotheken auf Basis von Linked Data Technologien [Elektronische Ressource] : Vortrag auf dem Bibliothekartag 2014 / Lydia Unterdörfel ; Björn Muschall. - Online-Ressource. - Stand: 02.06.2014. - 17 Folien (0,5 MB)
Adresse: http://amsl.technology/wp-content/uploads/2014/06/Pr%C3%A4sentation-Bibtag2014_Unterdoerfel.pdf
Gesehen: 18.06.2014

Using Existing Bibliographic Identifiers as Uniform Resource Names, RFC 2288 [Elektronische Ressource] / C. Lynch, C. Preston, R. Daniel - Online-Ressource. - Stand: Februar 1998. - 10 S.
Adresse: http://tools.ietf.org/html/rfc2288
Gesehen: 25.06.2014

Using International Standard Book Numbers as Uniform Resource Names, RFC 3187 [Elektronische Ressource] / J. Hakala, H. Walravens. - Online-Ressource. - Stand: Oktober 2001. - 11 S.
Adresse: http://tools.ietf.org/html/rfc3187
Gesehen: 25.06.2014

W3C Semantic Web Frequently Asked Questions [Elektronische Ressource] / Tim Berners-Lee, Dan Brickley, Eric Miller … - Online-Ressource. - Stand: 12.11.2009
Adresse: http://www.w3.org/RDF/FAQ
Gesehen: 05.05.2014

Was ist d:swarm? [Elektronische Ressource] / Jens Mittelbach. - Online-Ressource. - Stand: 30.05.2014
Adresse: http://dmp.slub-dresden.de/datenmanagement/dswarm-hilfe/was-ist-dswarm/
Gesehen: 06.06.2014

Web 3.0 [Interview with Jim Hendler] [Elektronische Ressource] : the Internet is changing … again / by Cade Metz. - Online Ressource. - 14.03.2007. - (PCMag)
Adresse: http://www.pcmag.com/article2/0,2817,2102864,00.asp
GEsehen: 15.06.2014

What is OntoWiki? What can it do for you? [Elektronische Ressource] / Arndt, Natanael. - Online-Ressource. - Stand: 24.04.2014
Adresse: https://github.com/AKSW/OntoWiki/wiki
Gesehen: 29.06.2014

Wohlgeformtheit, Gültigkeit und Vollständigkeit einer XML-Datei [Elektronische Ressource]. - Online-Ressource. - Stand: 16.04.2012. - (SelfHTML)
Adresse: http://de.selfhtml.org/xml/regeln/begriffe.htm
Gesehen: 13.06.2014

XML in 10 points [Elektronische Ressource] / Bert Bos. - Online Ressource. - Stand: 25.01.2011
Adresse: http://www.w3.org/XML/1999/XML-in-10-points.html.en
Gesehen: 02.06.2014

XML Schema annotation Element [Elektronische Ressource]. - Online-Ressource. - (w3schools)
Adresse: http://www.w3schools.com/schema/el_annotation.asp
Gesehen: 15.05.2014

XML Schema complexType Element [Elektronische Ressource]. - Online-Ressource. - (w3schools)
Adresse: http://www.w3schools.com/schema/el_complextype.asp
Gesehen: 15.05.2014

XML Schema documentation Element [Elektronische Ressource]. - Online-Ressource. - (w3schools)
Adresse: http://www.w3schools.com/schema/el_documentation.asp
Gesehen: 15.05.2014

XML Schema include Element [Elektronische Ressource]. - Online-Ressource. - (w3schools)
Adresse: http://www.w3schools.com/schema/el_include.asp
Gesehen: 15.05.2014

XML Schema restriction Element [Elektronische Ressource]. - Online-Ressource. - (w3schools)
Adresse: http://www.w3schools.com/schema/el_restriction.asp
Gesehen: 15.05.2014

XML Schema sequence Element [Elektronische Ressource]. - Online-Ressource. - (w3schools)
Adresse: http://www.w3schools.com/schema/el_sequence.asp
Gesehen: 15.05.2014

XSD Attributes [Elektronische Ressource]. - Online-Ressource. - (w3schools)
Adresse: http://www.w3schools.com/schema/schema_simple_attributes.asp
Gesehen: 15.05.2014

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Abschlussarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß veröffentlichten oder nicht veröffentlichten Quellen entnommen wurden, sind als solche kenntlich gemacht.

Leipzig, 04.07.2014 Annika Domin

Anhänge

Anhangsverzeichnis

Anhang A: Ressourcen des COUNTER-Vokabulars

Erläuterung

Der Anhang bietet eine Übersicht über alle im COUNTER-Vokabular definierten Klassen und Eigenschaften. Die im Zuge der `owl:OneOf`-Auflistungen ebenfalls definierten Individuen werden aufgrund ihrer Menge nachfolgend nicht aufgelistet. Sie sind der im elektronischen Anhang D enthaltenen Turtle-Datei *vocab_count.ttl* zu entnehmen.

Zu Anfang findet sich eine Übersicht der im Vokabular verwendeten Namensräume mitsamt der für diese im Turtle-Code genutzten Präfixe. Die Klassen sind schlicht alphabetisch nach ihrem Namen sortiert. Da die Eigenschaften bei einer einfachen alphabetischen Auflistung unübersichtlich wirken, sind diese zunächst ihren als `rdfs:domain` angegebenen Klassen nach geordnet und innerhalb dieses Systems alphabetisch sortiert.

Die Einträge bieten jeweils zunächst eine kurze Zusammenstellung der wichtigsten Fakten zu einer Ressource. Diese beginnt mit der über `rdf:type` zugewiesenen Klasse, gefolgt von dem deutschen und dem englischen Label. Falls vorhanden, schließen sich daran die Oberklasse aus dem COUNTER-Vokabular bzw. Domain und Range an. Danach folgt eine kurze Erläuterung der Ressource, die zumeist dem leicht erweiterten deutschen Kommentar entspricht. Bei geschlossenen Klassen werden daraufhin die Instanzen der Klasse aufgezählt, liegen Kardinalitätseinschränkungen vor, werden diese ebenfalls angegeben. Eigenschaften, die grundsätzlich in reifizierten Aussagen angewendet werden, ist dies angemerkt. Da in diesen Fällen keine Kardinalitätseinschränkungen modelliert werden konnten, ist bei der jeweiligen Eigenschaft ebenfalls angegeben, wie häufig sie pro Quellreport zu verwenden ist.

Übersicht

Verwendete Namensräume

- Eigener Namensraum für das COUNTER-Vokabular. Hier werden die Ressourcen des Vokabulars definiert.
  Präfix: `count`, URI: `http://www.platzhalter.de/counter#`

- Namensraum für Ressourcen des Resource Description Frameworks, die bei Definitionen für Grundlegendes wie Type und Property genutzt werden.
  Präfix: `rdf`, URI: `http://www.w3.org/1999/02/22-rdf-syntax-ns#`

- Namensraum für Ressourcen von RDF-Schema, die für etliche Definitionen genutzt werden, wie etwa von Klassen und Unterklassen, Domain und Range oder Kommentaren und Labels.
  Präfix: `rdfs`, URI: `http://www.w3.org/2000/01/rdf-schema#`

- Namensraum der Ressourcen für die Web Ontology Language, die zur Definition von Klassen und Collections genutzt wird.
  Präfix: `owl`, URI: `http://www.w3.org/2002/07/owl#`

- Namensraum der Ressourcen für XML-Schema, die bei einigen Eigenschaften in Form einfacher Datentypen als Range angegeben werden.
  Präfix: `xsd`, URI: `http://www.w3c.org/2001/XMLSchema#`

- Namensraum für Ressourcen des Dublin Cores, die lediglich für die Beschreibung der Metadaten des COUNTER-Vokabulars genutzt werden.
  Präfix: `dc`, URI: `http://purl.org/dc/elements/1.1/`

Klassen

count:Category

> Typ..... owl:Class
> Label, dt. ..... Kategorie
> Label, engl..... category

Klasse der Kategorien der in einer Zähleinheit gemessenen Nutzung.
Geschlossene Klasse mit drei Instanzen:
count:Requests, count:Searches und count:AccessDenied.

count:Consortium

> Typ..... rdfs:Class
> Label, dt. ..... Konsortium
> Label, engl..... consortium

Klasse der Konsortien, an denen Kunden beteiligt sind. Bei Konsortialreports genutzt.

count:Contact

> Typ..... rdfs:Class
> Label, dt. ..... Kontakt
> Label, engl..... contact

Klasse der Kontaktmöglichkeiten mit Kontaktdaten. Obwohl im XML-Schema ausdrücklich als „Person“ deklariert, wird in vielen Reports ein allgemeiner Support-Kontakt angegeben. Entsprechend hier als Kontaktmöglichkeit modelliert, muss keine Person sein.
Kardinalitätseinschränkungen für folgende Eigenschaften:
count:hasEmail 0≤c≤1.

count:CountingInstance

> Typ..... rdfs:Class
> Label, dt. ..... Zählinstanz
> Label, engl..... counting instance

Klasse der Zählinstanzen. Dies sind abstrakte Vorstellungen von Zähleinheiten, die die Zählung einer bestimmten Nutzungskategorie und Messgröße für einen Monat bei einem Reportelement sowie das Zählergebnis umfassen.
Kardinalitätseinschränkungen für folgende Eigenschaften:
count:measuredForPeriod c=1, count:hasCategory c=1, count:hasMetricType c=1, count:hasCount =1, count:considersPubYear 0≤c≤1, count:wasCountedIn c=1, count:refersToCustomer x=1.

count:Customer

Typ..... `rdfs:Class`
Label, dt. ..... `Kunde`
Label, engl..... `customer`
Oberklasse ..... `count:Organization`

Klasse der Kunden, über deren Daten die Reports berichten. Nur Organisationen können Kunden sein.
Kardinalitätseinschränkungen für folgende Eigenschaften:
`count:recieves c>1`

count:DateRange

Typ..... `rdfs:Class`
Label, dt. ..... `Zeitraum`
Label, engl..... `date range`

Klasse der Zeiträume. Entweder der Messzeitraum, welcher immer einen Kalendermonat umfasst, oder der in Journal Report 5 berücksichtigte Zeitraum von Publikationsjahren einer Zeitschrift. Hier beginnt der Zeitraum am 1. Januar des ersten berücksichtigten Publikationsjahres und endet am 31.Dezember des letzten.
Kardinalitätseinschränkungen für folgende Eigenschaften:
`count:hasStartDay c=1, count:hasEndDay c=1.`

count:ItemDatatype

Typ..... `owl:Class`
Label, dt. ..... `Medienart`
Label, engl..... `item data type`

Klasse der Datentypen von Reportelementen. Datentypen werden zur Beschreibung des Mediums des Elements genutzt.
Geschlossene Klasse mit sechs Instanzen:
`count:Journal, count:Database, count:Platform, count:Book, count:Collection, count:Multimedia`

count:MetricType

Typ..... `owl:Class`
Label, dt. ..... `Messgröße`
Label, engl..... `metric type`

Klasse der zählbaren Messgrößen. Diese definieren, welche Art von Nutzung gemessen wird, beispielsweise Volltextabrufe im HTML-Format oder Abweisungen aufgrund von fehlenden Lizenzen. Geschlossene Klasse mit 25 Instanzen:
`count:FullTextPostscript, count:FullTextPostscriptMobile, count:FullTextPdf, count:FullTextPdfMobile, count:FullTextHtml, count:FullTextHtmlMobile,`

count:FullTextEpub, count:SectionedHtml, count:FullTextTotal, count:ToC, count:Abstract, count:Reference, count:DataSet, count:Audio, count:Video, count:Image, count:Podcast, count:Multimedia, count:RecordView, count:ResultClick, count:SeachRegular, count:SearchFederated, count:Turnaway, count:NoLicense, count:Other

count:Organization

Typ..... rdfs:Class
Label, dt. ..... Organisation
Label, engl..... organization
Unterklassen ..... count:Customer, count:Vendor

Klasse der Organisationen. Keine Instanzen dieser Klasse vorgesehen, sondern reine Oberklasse von Vendor und Customer.

count:Platform

Typ..... rdfs:Class
Label, dt. ..... Plattform
Label, engl..... platform

Klasse der Plattformen, auf denen die lizenzierten Daten zur Nutzung angeboten werden. Bezeichnung durch den Namen dieses Online-Hosts in der durch den Serviceanbieter genutzten Form. Beispiel: EBSCOhost.

count:Publisher

Typ..... rdfs:Class
Label, dt. ..... Verleger
Label, engl..... publisher

Klasse der Verleger von Reportelementen.

count:Report

Typ..... rdfs:Class
Label, dt. ..... Report
Label, engl..... report

Klasse der einzelnen COUNTER-Reports.
Kardinalitätseinschränkungen für folgende Eigenschaften:
count:wasCreatedOn $0 \leq c \leq 1$, count:hasReportTitle $0 \leq c \leq 1$, count:hasReportVersion $0 \leq c \leq 1$, count:hasReportID $c=1$.

count:ReportItem

> Typ..... rdfs:Class
> Label, dt. ..... Reportelement
> Label, engl..... report item

Klasse der Reportelemente. Dies sind Repräsentationen der einzelnen Ressourcen, über deren Nutzung im Report berichtet wird. Reportelemente sind eine Art Spiegelbild der echten Ressourcen, die nur auf Grundlage von Daten aus COUNTER-Reports entstehen. Anknüpfungspunkt zu den anderen Teilen des ERM.
Kardinalitätseinschränkungen für folgende Eigenschaften:
`count:hasItemDatatype c=1, count:hasPerformance 1≤c, count:IsContainedIn 1≤c, count:CounterRepresentedMedium c=1.`

count:ReportTitle

> Typ..... owl:Class
> Label, dt. ..... Reportbezeichnung
> Label, engl..... report title

Klasse der Bezeichnungen der Reports. Explizit nicht die Reports selbst, sondern deren den Inhalt bestimmenden Bezeichnungen.
Geschlossene Klasse mit 26 Instanzen:
`count:BR1, count:BR2, count:BR3, count:BR4, count:BR5, count:CR1, count:CR2, count:CR3, count:DB1,count:DB2, count:JR1, count:JR1GOA, count:JR1a, count:JR2, count:JR3, count:JR3mobile, count:JR4, count:JR5, count:MR1, count:MR2, count:PR1, count:TR1, count:TR1mobile, count:TR2, count:TR3, count:TR3mobile.`

count:Vendor

> Typ..... rdfs:Class
> Label, dt. ..... Datenanbieter
> Label, engl..... vendor
> Oberklasse ..... count:Organization

Klasse der Datenanbieter, die die Reports erstellen. Nur Organisationen können Datenanbieter sein.

Eigenschaften

Consortium

count:hasConsortiumCode

> Typ..... `owl:DatatypeProperty`
> Label, dt. ..... `hat den Code`
> Label, engl..... `has the code`
> Domain ..... `count:Consortium`
> Range..... `xsd:string`

Verknüpft die Abkürzung oder den Code, der das Konsortium identifiziert. Anzuwenden in reifizierter Aussage unter Angabe des Quellreports. Höchstens einmal pro Report zu verwenden.

count:hasConsortiumName

> Typ..... `owl:DatatypeProperty`
> Label, dt. ..... `ist bekannt unter dem Namen`
> Label, engl..... `has the well known name`
> Domain ..... `count:Consortium`
> Range..... `xsd:string`

Verknüpft den Namen, unter dem das Konsortium bekannt ist. Anzuwenden in reifizierter Aussage unter Angabe des Quellreports. Genau einmal pro Report zu verwenden.

Contact

count:hasContactName

> Typ..... `owl:DatatypeProperty`
> Label, dt. ..... `hat den Namen`
> Label, engl..... `has the name`
> Domain ..... `count:Contact`
> Range..... `xsd:string`

Verknüpft den Namen des Kontaktes bzw. der Kontaktperson. Anzuwenden in reifizierter Aussage unter Angabe des Quellreports. Höchstens einmal pro Report zu verwenden.

count:hasEmail

> Typ..... `owl:ObjectProperty`
> Label, dt. ..... `hat die E-Mail-Adresse`
> Label, engl..... `has the email adress`
> Domain ..... `count:Contact`

Verknüpft die E-Mail-Adresse des Kontaktes bzw. der Kontaktperson.

CountingInstance

count:considersPubYear

> Typ..... owl:ObjectProperty
> Label, dt. ..... berücksichtigt den Publikationszeitraum
> Label, engl..... considers the publication year range
> Domain ..... count:CountingInstance
> Range..... count:DateRange

Verknüpft in Journal Report 5 den Zeitraum von Publikationsjahren einer Zeitschrift, der bei der Nutzungsmessung berücksichtigt wurde. Dieser kann ein oder mehrere Kalenderjahre umfassen.

count:hasCount

> Typ..... owl:DatatypeProperty
> Label, dt. ..... hat den Zählwert
> Label, engl..... has the count
> Domain ..... count:CountingInstance
> Range..... xsd:nonNegativeInteger

Verknüpft den konkreten Zählwert.

count:hasMetricType

> Typ..... owl:ObjectProperty
> Label, dt. ..... hat die Messgröße
> Label, engl..... has the metric type
> Domain ..... count:CountingInstance
> Range..... count:MetricType

Verknüpft die gezählte Messgröße. Zulässige Objektressourcen sind vordefiniert, siehe Klasse `count:MetricType`.

count:measuredForPeriod

> Typ..... owl:ObjectProperty
> Label, dt. ..... gemessen für den Zeitraum
> Label, engl..... was measured for the period
> Domain ..... count:CountingInstance
> Range..... count:DateRange

Verknüpft den Zeitabschnitt, während dem die Nutzung gemessen wurde. Dieser umfasst genau einen Kalendermonat.

count:measuresCategory

> Typ..... owl:ObjectProperty
> Label, dt. ..... misst die Kategorie
> Label, engl..... measures category
> Domain ..... count:CountingInstance
> Range..... count:Category

Verknüpft die in dieser Zähleinheit gemessene Nutzungskategorie. Zulässige Objektressourcen sind vordefiniert, siehe Klasse `count:Category`.

count:refersToCustomer

> Typ..... owl:ObjectProperty
> Label, dt. ..... bezieht sich auf den Kunden
> Label, engl..... refers to the customer
> Domain ..... count:CountingInstance
> Range..... count:Customer

Verknüpft den Kunden, auf den sich die gemessenen Nutzungsdaten beziehen.
Wichtig wegen der Möglichkeit mehrerer Kunden in einem Report.

count:wasCountedIn

> Typ..... owl:ObjectProperty
> Label, dt. ..... wurde gezählt in
> Label, engl..... was counted in
> Domain ..... count:CountingInstance
> Range..... count:Report

Verknüpft den diese Zählinstanz beinhaltenden COUNTER Report.

Customer

count:hasInstitutionalIdentifier

> Typ..... rdf:Property
> Label, dt. ..... hat Insititutions-Identifikator
> Label, engl..... has institutional identifier
> Domain ..... count:Customer

Soll die Standardidentifikatoren verknüpfen, die den Kunden identifizieren.
Diese sind theoretisch vordefiniert, momentan existieren aber noch keine entsprechend standardisierten Identifikatoren. Daher wird diese Property noch nicht eingesetzt. Wenn entsprechende Identifikatoren anerkannt werden, sollte, analog zu der Umsetzung von ItemIdentifiern wie DOI und ISBN, für jeden Identifikator eine eigene Property definiert werden.

count:isPartOfConsortium

> Typ..... owl:ObjectProperty
> Label, dt. ..... ist Teil des Konsortiums
> Label, engl..... is part of consortium
> Domain ..... count:Customer
> Range..... count:Consortium

Verknüpft das Konsortium, an dem der Kunde beteiligt ist. Für Konsortialreports verwendet. Anzuwenden in reifizierter Aussage unter Angabe des Quellreports. Höchstens einmal pro Report zu verwenden.

count:receives

> Typ..... owl:ObjectProperty
> Label, dt. ..... erhält
> Label, engl..... receives
> Domain ..... count:Customer
> Range..... count:Report

Verknüpft den COUNTER Report, den der Kunde erhält.

DateRange

count:hasEndDay

> Typ..... owl:DatatypeProperty
> Label, dt. ..... endet am Tag
> Label, engl..... ends on day
> Domain ..... count:DateRange
> Range..... xsd:date

Verknüpft den letzten Tag eines Zeitraumes.

count:hasStartDay

> Typ..... owl:DadatypeProperty
> Label, dt. ..... beginnt am Tag
> Label, engl..... begins on day
> Domain ..... count:DateRange
> Range..... xsd:date

Verknüpft den ersten Tag eines Zeitraumes

Organization

count:hasContact

Typ..... owl:ObjectProperty
Label, dt. ..... hat den Kontakt
Label, engl..... has the contact
Domain ..... count:Organization
Range..... count:Contact

Verknüpft den Kontakt bzw. die Kontaktperson in einer Organisation. Anzuwenden in reifizierter Aussage unter Angabe des Quellreports.

count:hasLogoURL

Typ..... owl:ObjectProperty
Label, dt. ..... hat die Logo-URL
Label, engl..... has the logo URL
Domain ..... count:Organization

Verknüpft die URL eines Organisations-Logos. Anzuwenden in reifizierter Aussage unter Angabe des Quellreports. Höchstens einmal pro Report zu verwenden.

count:hasOrganizationID

Typ..... owl:DatatypeProperty
Label, dt. ..... hat die Organisations-ID
Label, engl..... has the organization ID
Domain ..... count:Organization
Range..... xsd:string

Verknüpft einen Identifikator, unter dem eine Organisation dem Inhaltsanbieter bekannt ist, in der von dem Anbieter intern genutzten Form. Anzuwenden in reifizierter Aussage unter Angabe des Quellreports. Genau einmal pro Report zu verwenden.

count:hasOrganizationName

Typ..... owl:DatatypeProperty
Label, dt. ..... hat den Organisationsamen
Label, engl..... has the organization name
Domain ..... count:Organization
Range..... xsd:string

Verknüpft den Namen einer Organisation. Anzuwenden in reifizierter Aussage unter Angabe des Quellreports. Höchstens einmal pro Report zu verwenden.

count:hasWebsiteURL

> Typ ..... `owl:ObjectProperty`
> Label, dt. ..... `hat die Website-URL`
> Label, engl. ..... `has the website URL`
> Domain ..... `count:Organization`

Verknüpft die URL einer Organisations-Website. Anzuwenden in reifizierter Aussage unter Angabe des Quellreports. Höchstens einmal pro Report zu verwenden.

Report

count:hasReportID

> Typ ..... `owl:DatatypeProperty`
> Label, dt. ..... `hat die Report-ID`
> Label, engl. ..... `has the report ID`
> Domain ..... `count:Report`
> Range ..... `xsd:string`

Verknüpft eine von der die Nachricht erstellenden Anwendung zu Trackingzwecken vergebene Report-ID.

count:hasReportTitle

> Typ ..... `owl:ObjectProperty`
> Label, dt. ..... `hat die Report-Bezeichnung`
> Label, engl. ..... `has the report title`
> Domain ..... `count:Report`
> Range ..... `count:ReportTitle`

Verknüpft die Bezeichnung des Reports. Zulässige Objektressourcen sind vordefiniert, siehe Klasse `count:ReportTitle`.

count:hasReportVersion

> Typ ..... `owl:DatatypeProperty`
> Label, dt. ..... `hat die Reportversion`
> Label, engl. ..... `has the report version`
> Domain ..... `count:Report`
> Range ..... `xsd:string`

Verknüpft die Version des COUNTER-Reports..

count:wasCreatedOn

> Typ..... owl:DatatypeProperty
> Label, dt. ..... wurde erstellt am
> Label, engl..... was created on
> Domain ..... count:Report
> Range..... xsd:dateTime

Verknüpft Datum und Uhrzeit, zu der der Report erstellt wurde.

ReportItem

count:CounterRepresentedMedium

> Typ..... owl:ObjectProperty
> Label, dt. ..... durch das Reportelement repräsentiertes Medium
> Label, engl..... medium represented by the report item
> Domain ..... count:ReportItem

Verknüpft das Medium, das durch das Reportelement repräsentiert wird, um seine Nutzung zu berichten. Anknüpfungspunkt an die anderen Teile des ERMS. Von der UB Leipzig vorerst nicht genutzt, da das Gesamtsystem bislang nur Zeitschriften umfasst. Anknüpfungspunkt wird also die angegebene Online-ISSN.

count:hasDOI

> Typ..... owl:ObjectProperty
> Label, dt. ..... hat den DOI
> Label, engl..... has the DOI
> Domain ..... count:ReportItem

Verknüpft den DOI einer Ressource.

count:hasItemDatatype

> Typ..... owl:ObjectProperty
> Label, dt. ..... hat den Element-Datentyp
> Label, engl..... has the item data type
> Domain ..... count:ReportItem
> Range..... count:ItemDatatype

Verknüpft den Datentyp des Reportelements, der die Art des Mediums beschreibt. Zulässige Objektressourcen sind vordefiniert, siehe Klasse `count:ItemDatatype`.

count:hasItemName

Typ	owl:DatatypeProperty
Label, dt.	hat den Namen
Label, engl.	has the item name
Domain	count:ReportItem
Range	xsd:string

Verknüpft die Bezeichnung des Elementes, z. B. den Titel der Zeitschrift. Anzuwenden in reifizierter Aussage unter Angabe des Quellreports. Genau einmal pro Report zu verwenden.

count:hasLicensee

Typ	owl:ObjectProperty
Label, dt.	hat den Lizenznehmer
Label, engl.	has the licensee
Domain	count:ReportItem
Range	count:Customer

Verknüpft den Lizenznehmer (Kunden) mit einem spezifischen Element des Reports. Anzuwenden in reifizierter Aussage unter Angabe des Quellreports. Mindestens einmal pro Report zu verwenden, mehrere Kunden können beispielsweise in Konsortialreports verknüpft werden.

count:hasLicensor

Typ	owl:ObjectProperty
Label, dt.	hat die Lizenzgeber
Label, engl.	has the licensor
Domain	count:ReportItem
Range	count:Vendor

Verknüpft den Lizenzgeber (Datenanbieter) mit einem spezifischen Element des Reports. Anzuwenden in reifizierter Aussage unter Angabe des Quellreports. Genau einmal pro Report zu verwenden.

count:hasOnlineISBN

Typ	owl:ObjectProperty
Label, dt.	hat die Online-ISBN
Label, engl.	has the online ISBN
Domain	count:ReportItem

Verknüpft die ISBN der digitalen Ausgabe eines Buches.
Probleme: Nach Empfehlungen der ISBN Agency sollten E-Books verschiedener Formate analog zu gedruckten Büchern verschiedener Ausgaben jeweils eine individuelle ISBN zuge-

ordnet werden. Dies wird jedoch nur von einem Teil der Verleger praktiziert, wodurch teilweise übergeordnete ISBNs für alle Formate eines E-Books und teils individuelle ISBNs für jedes einzelne Format existieren.[1] COUNTER sieht keine Formattrennung auf Ebene der Reportelemente und somit der ISBN vor, sondern erst bei der Messung einer bestimmten Messgröße in der Zähleinheit. Bei E-Books mit nach Formaten getrennten ISBN müssten also entweder für jedes Format eigene ReportItems erstellt werden, wodurch eine eindeutige Zuordnung von Format und ISBN gegeben wäre. Alternativ müssten alle ISBNs für alle Formate ohne eindeutige Zuordnung mit einem einzigen Reportelement verknüpft werden. Dies liegt im Ermessen des den Report erstellenden Datenanbieters.

count:hasOnlineISSN

Typ ..... `owl:ObjectProperty`
Label, dt. ..... `hat die Online-ISSN`
Label, engl. ..... `has the online ISSN`
Domain ..... `count:ReportItem`

Verknüpft die ISSN der digitalen Ausgabe einer Schriftenreihe. In der UB Leipzig vorerst Anknüpfungspunkt zum Gesamtsystem, da bisher nur Zeitschriften verzeichnet werden.

count:hasPerformance

Typ ..... `owl:ObjectProperty`
Label, dt. ..... `hat die Leistung`
Label, engl. ..... `has the performance`
Domain ..... `count:ReportItem`
Range ..... `count:CountingInstance`

Verknüpft die Statistiken, die jeweils eine Nutzungskategorie und eine Messgröße für den Zeitraum von einem Monat abbilden.

count:hasPrintISBN

Typ ..... `owl:ObjectProperty`
Label, dt. ..... `hat die Print-ISBN`
Label, engl. ..... `has the print ISBN`
Domain ..... `count:ReportItem`

Verknüpft die ISBN der gedruckten Ausgabe eines Buches.
Probleme:
Bei Vorhandensein von inhaltsgleichen Printausgaben verschiedener Formate (z. B. Hardcover und Paperback) können diese unterschiedliche ISBNs besitzen. Ob diese ISBNs alle angegeben werden und wenn nicht, welche auszuwählen ist, wird von COUNTER nicht geregelt und liegt im Ermessen des den Report erstellenden Datenanbieters.

[1] Vgl. E-Books and ISBNs [Elektronische Ressource].

count:hasPrintISSN

> Typ ..... `owl:ObjectProperty`
> Label, dt. ..... `hat die Print-ISSN`
> Label, engl. ..... `has the print ISSN`
> Domain ..... `count:ReportItem`

Verknüpft die ISSN der Printausgabe einer Schriftenreihe.

count:hasProprietaryID

> Typ ..... `owl:DatatypeProperty`
> Label, dt. ..... `hat die proprietäre ID`
> Label, engl. ..... `has the proprietary ID`
> Domain ..... `count:ReportItem`
> Range ..... `xsd:string`

Verknüpft die proprietäre ID einer Ressource, wie sie intern vom Datenanbieter verwendet wird.

count:isAccessibleVia

> Typ ..... `owl:ObjectProperty`
> Label, dt. ..... `ist zugänglich über`
> Label, engl. ..... `is accessible via`
> Domain ..... `count:ReportItem`
> Range ..... `count:Platform`

Verknüpft die Plattform. Anzuwenden in reifizierter Aussage unter Angabe des Quellreports. Genau einmal pro Report zu verwenden.

count:isContainedIn

> Typ ..... `owl:ObjectProperty`
> Label, dt. ..... `ist enthalten in`
> Label, engl. ..... `is contained in`
> Domain ..... `count:ReportItem`
> Range ..... `count:Report`

Verknüpft COUNTER Report, der das Reportelement enthält.

count:wasPublishedBy

> Typ..... owl:ObjectProperty
> Label, dt. ..... wurde publiziert von
> Label, engl..... was published by
> Domain ..... count:ReportItem
> Range..... count:Publisher

Verknüpft den Verleger eines Reportelements. Anzuwenden in reifizierter Aussage unter Angabe des Quellreports. Höchstens einmal pro Report zu verwenden.

ReportTitle

count:hasFullTitle

> Typ..... owl:DatatypeProperty
> Label, dt. ..... hat die Vollbezeichnung
> Label, engl..... has the full title
> Domain ..... count:ReportTitle
> Range..... xsd:string

Verknüpft den Volltitel eines COUNTER-Reports, z. B. Journal Report 1.

Verschiedenes

count:hasSourceReport

> Typ..... owl:ObjectProperty
> Label, dt. ..... stammt aus dem Quellreport
> Label, engl..... was taken from the source report
> Range..... count:Report

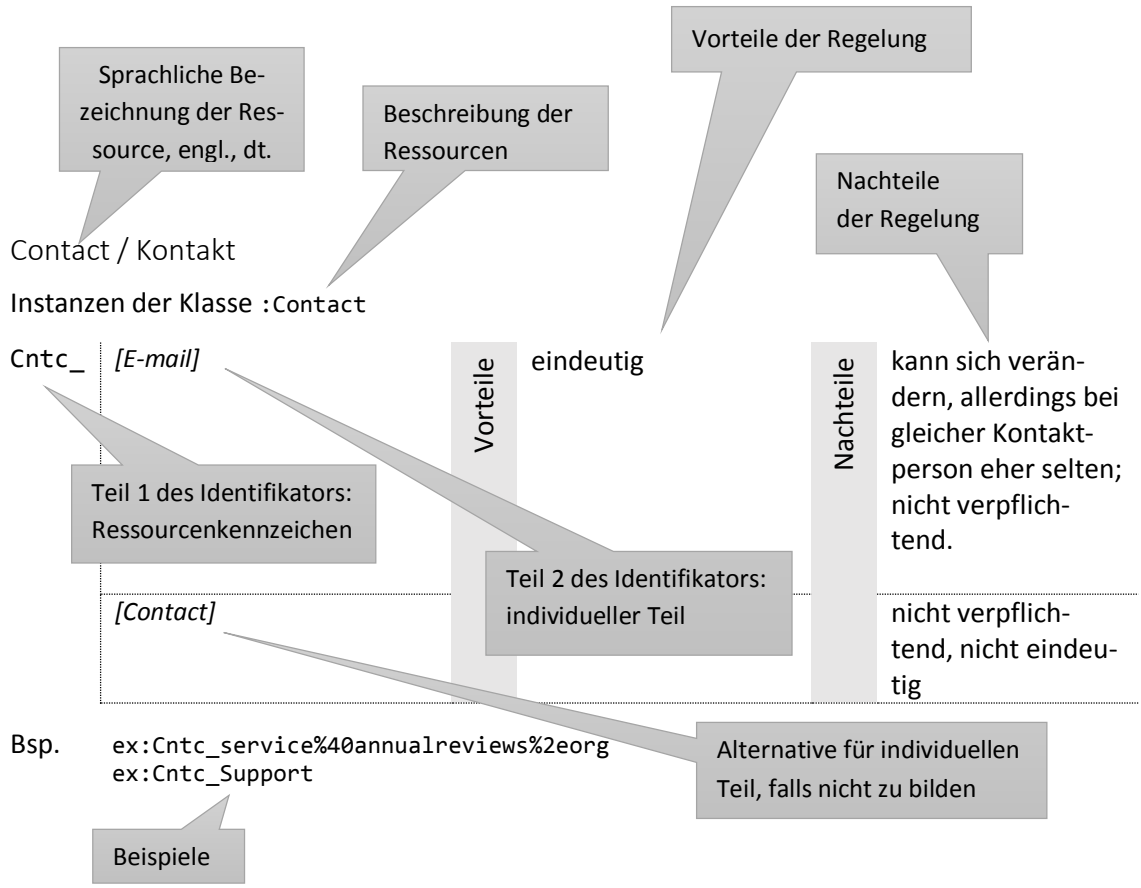
Verknüpft den Quellreport, aus dem die Informationen entnommen wurden, mit den jeweiligen Daten.

Anhang B: URI-Vergabe

Erläuterung

Die Regelungen zur URI-Vergabe sind für jede als Individual zu modellierende Ressourcenart tabellarisch in einzelnen Absätzen aufgeführt. Diese Ressourcen sind alphabetisch nach ihrer sprachlichen Bezeichnung sortiert.

Schematische Erklärung des Aufbaus einer Tabelle:



Im individuellen Teil des Identifikators werden Platzhalter in eckigen Klammern verwendet. Die eckigen Klammern sind nicht Teil des Identifikators, sondern Teil des Platzhalters. Platzhalter können drei Arten von Inhalten vertreten.

1. Sie stehen direkt für den Inhalt des XML-Elements bzw. -Attributs, dessen Namen sie tragen. Gibt es in einem Report je nach Schachtelung mehrere Elemente des gleichen Namens, ist das kontextuell passende gemeint.
2. Sie beziehen sich auf eine andere RDF-Ressource. in diesem Fall steht in runden Klammern `(RDF)` hinter dem Namen des Platzhalters. Diese Art Platzhalter steht für den individuellen Teil des Identifikators der Ressource und wird durch diesen ersetzt.
3. Sie vertreten laufende Nummern. Bei der Ersetzung der Platzhalter durch Werte aus der XML-Datei muss wenn nötig die Prozent-Kodierung für URIs angewendet werden.

Identifikatoren, die bereits gültige URIs sind, werden in spitzen Klammern geschrieben <>. Allen anderen muss noch der globale Teil des URI bzw. das Präfix des qnames vorangestellt werden. In den Beispielen wird das Präfix ex: für den example-Namespace verwendet.

Beispiele:

`Cstm_`*[ID]*`_`*[Name]*

⇨ `ex:Cstm_26838_Universitaet%20Leipzig`

`Pltf_`*[Vendor (RDF)]*`_`*[ItemPlatform]*

⇨ `ex:Pltf_Ovid_Ovid%20Technologies%20Inc%2e_Ovid`

`Rprt_`*[#laufende Nummer]*

⇨ `ex:Rprt_42`

`<mailto:`*[E-mail]*`>`

⇨ `<support@ovid.org>`.

Übersicht

Consortium / Konsortium

Instanzen der Klasse `:Consortium`

`Cnsr_`	*[WellKnownName]*	Vorteile	verpflichtend anzugeben	Nachteile	Der WellKnown-Name ist nur definiert als „ein Name, unter dem das Konsortium bekannt ist“, mehrere parallel existierende Namen könnten zu mehreren Ressourcen für ein einzelnes Konsortium führen.

Bsp.
```
ex:Cnsr_Consortium%20of%20Swiss%20Academic%20Libraries
ex:Cnsr_Niedersachsen%20Konsortium
ex:Cnsr_Consortium%20universitaire%20de%20publications%20num%c3%a9
riques
```

Contact / Kontakt

Instanzen der Klasse `:Contact`

`Cntc_`	*[E-mail]*	Vorteile	eindeutig	Nachteile	kann sich verändern, allerdings bei gleicher Kontaktperson eher selten; nicht verpflichtend.
	[Contact]				nicht verpflichtend, nicht eindeutig

Bsp.
```
ex:Cntc_service%40annualreviews%2eorg
ex:Cntc_Support
```

Contact e-mail / E-Mail-Adresse des Kontakts

Mit `:hasEmail` als Objekt verknüpfte Ressourcen

`<mailto:`*[E-mail]*`>`		Vorteile	ist bereits schemakonformer URI	Nachteile	

Bsp.
```
<mailto: service@annualreviews.org>
<mailto: support@services.acs.org>
```

Counting Instance / Zählinstanz

Instanzen der Klasse `:CountingInstance`

`CntI_`	*[#laufende Nummer]*	Vorteile	eindeutig ; möglich, da einmalige Information	Nachteile	

Bsp. `ex:CntI_42`

Customer / Kunde

Instanzen der Klasse `:Customer`

`Cstm_`	*[ID]_[Name]*	Vorteile	mit hoher Wahrscheinlichkeit eindeutig	Nachteile	Name nicht verpflichtend anzugeben
	[ID]		verpflichtend anzugeben. Bei vielen Reports Identifikation über Webiste-URL, diese ist global eindeutig.		ID wird vom Datenanbieter vergeben und ist daher nicht global eindeutig. Größere Wahrscheinlichkeit des doppelten Auftretens, als wenn noch zusätzlich durch Namen identifiziert

Bsp.
```
ex:Cstm_2360923_UNIV%20LEIPZIG
ex:Cstm_135775_WIB6091%20-%20Universitaetsklinikum%20Leipzig
```

Date range / Zeitraum

Instanzen der Klasse `:DateRange`

DtRg_	*[Begin]_[End]* *[PubYr]*-01-01_*[PubYr]*-12-31 *[PubYrFrom]*-01-01_*[PubYrTo]*-12-31	Vorteile	alles verpflichtend anzugeben ; eindeutig	Nachteile

Bsp.
```
ex:DtRg_2013-07-01_2013-07-31
ex:DtRg_2011_01_01_2011-12-31
ex:DtRg_2005-01-01_2007-12-31
```

DOI of a report element / DOI eines Reportelements

Mit `:hasDOI` als Objekt verknüpfte Ressourcen

<http://doi.org/*[Value]*>	Vorteile	ist bereits schemakonformer URI	Nachteile

Bsp.
```
<http://doi.org/10%2e1162%2fafar%2e1000>
<http://doi.org/10%2e1146%2fanimal>
```

Logo URL / Logo-URL

Mit `:hasLogoURL` als Objekt verknüpfte Ressourcen

<*[LogoUrl]*>	Vorteile	ist bereits schemakonformer URI	Nachteile

Bsp.
```
<http://example.org/logo.jpg>
<http://www.example.com/logo2.png>
```

Online ISBN of a report element / Online-ISBN eines Reportelements

Mit `:hasOnlineISBN` als Objekt verknüpfte Ressourcen

<urn:isbn:*[Value]*>	Vorteile	ist bereits schemakonformer URI	Nachteile

Bsp.
```
<urn:ISBN:978-0-8412-2568-8>
<urn:ISBN:978-0-8412-1343-2>
```

Online ISSN of a report element / Online-ISSN eines Reportelements

Mit `:hasOnlineISSN` als Objekt verknüpfte Ressourcen

`<urn:issn:`*[Value]*`>`	Vorteile	ist bereits schemakonformer URI	Nachteile		

Bsp. `<urn:ISSN:2165-8110>`
`<urn:ISSN:1937-2108>`

Platform / Plattform

Instanzen der Klasse `:Platform`

Pltf_	*[Vendor (RDF)]_[ItemPlatform]*	Vorteile	durch Kombination mit Datenanbieter mit hoher Wahrscheinlichkeit eindeutig	Nachteile	bei Änderung des Namens wird zweite Ressource für eine einzelne Plattform angelegt

Bsp. `ex:Plfm_123ACS_American%20Chemical%20Society_ACS%20Publications`
`ex: Pltf_Ovid_Ovid%20Technologies%20Inc%2e_Ovid`

Print ISBN of a report element / Print-ISBN eines Reportelements

Mit `:hasPrintISBN` als Objekt verknüpfte Ressourcen

`<urn:isbn:`*[Value]*`>`	Vorteile	ist bereits schemakonformer URI	Nachteile		

Bsp. `<urn:isbn:978-0-8412-2567-1>`
`<urn:isbn:978-0-8412-2218-2>`

Print ISSN of a report element / Print-ISSN eines Reportelements

Mit `:hasPrintISSN` als Objekt verknüpfte Ressourcen

`<urn:issn:`*[Value]*`>`	Vorteile	ist bereits schemakonformer URI	Nachteile		

Bsp. `<urn:issn:2165-8102>`
`<urn:issn:0001-9933>`

Publisher / Verleger

Instanzen der Klasse `:Publisher`

Pbls_	*[ItemPublisher]*	Vorteile		Nachteile	Nur mit großer Wahrscheinlichkeit eindeutig, abhängig von Angaben des Datenanbieters; bei Änderung des Namens wird zweite Ressource für einen einzelnen Verleger angelegt

Bsp.
```
ex:Pbls_Annual%20Reviews
ex:Pbls_CAB%20International
ex:Pbls_MIT%20Press
```

Report / Report

Instanzen der Klasse `:Report`

Rprt_	*[#laufende Nummer]*	Vorteile	eindeutig ; möglich, da einmalige Information	Nachteile	

Bsp.
```
ex:Rprt_42
```

Report item / Reportelement

Instanzen der Klasse `:ReportItem`

RpIt_	doi*[DOI > Value]*	Vorteile	bereits von sich aus als eindeutiger Identifier konzipiert	Nachteile	nicht verpflichtend anzugeben
	isbn*[OnlineISBN > Value]* bzw. issn*[OnlineISSN > Value]*		ebenfalls eindeutiger Identifier		nicht verpflichtend anzugeben, nicht für alle Medienarten vorgesehen, Einbußen bei der Einheitlichkeit
	[ItemName]_[Vendor (RDF)]		Titel bei einem Vendor mit hoher Wahrscheinlichkeit eindeutig, beides verpflichtend anzugeben		beides veränderlich

Bsp.
```
ex:RpIt_doi10%2e1021%2fachre4
ex:RpIt_issn1234-5678
ex:RpIt_CAB%20Abstracts_Ovid%20Technologies%20Inc%2e
```

Vendor / Datenanbieter

Instanzen der Klasse `:Vendor`

Vndr_		Vorteile		Nachteile	
	[ID]_[Name]		mit hoher Wahrscheinlichkeit eindeutig		Name nicht verpflichtend anzugeben
	[ID]		verpflichtend anzugeben. Bei vielen Reports Identifikation über Webiste-URL, diese ist global eindeutig.		ID wird vom Datenanbieter selbst vergeben und ist daher nicht global eindeutig. Größere Wahrscheinlichkeit des doppelten Auftretens, als wenn noch zusätzlich durch Namen identifiziert

Bsp.
```
ex:Vndr_http%3a%2f%2fpubs%2eacs%2eorg_American%20Chemical%20Society
ex:Vndr_Ovid_Ovid%20Technologies%20Inc%2e
ex:Vndr_http%3a%2f%2fwww%2emitpressjournals%2eorg
```

Website URL / Website-URL

Mit `:hasWebsiteURL` als Objekt verknüpfte Ressourcen

	Vorteile		Nachteile
<*[WebsiteUrl]*>		ist bereits schemakonformer URI	

Bsp.
```
<http://pubs.acs.org>
<http://www.annualreviews.org>
```

Anhang C: XML-RDF-Konkordanz

Erläuterung

Die nachfolgende Tabelle stellt eine XML- und eine RDF-Reportdatei elementweise nebeneinander. In der linken Spalte sind zeilenweise alle Elemente eines XML-Reports in der richtigen Reihenfolge ihres Auftretens aufgeführt. Nacheinander geschrieben ergeben diese eine gültige XML-Datei nach dem COUNTER XML-Schema. Die mittlere Spalte zeigt die jeweiligen RDF-Entsprechungen in Turtle-Serialisierung und ergibt insgesamt eine vollständige Turtle-Datei. Die rechte Spalte enthält Anmerkungen und Ergänzungen.

Die Tabelle ist so aufgebaut, dass ein sequenzielles Übertragen der XML-Elemente nach RDF möglich ist. Daher kann sie als Grundlage für die Entwicklung des geplanten XML/RDF-Konverters genutzt werden.

Die Variablennamen für die Elementwerte symbolisieren den für diese jeweils geforderten Datentyp und sind innerhalb eines solchen Datentyps laufend durchnummeriert.

Von Variablen angezeigte Datentypen:

aUri ..... `xsd:anyURI`,

date..... `xsd:date`,

dtTm ..... `xsd:dateTime`,

Intg..... `xsd:Integer`,

strg ..... `xsd:string`,

year..... `xsd:year`,

vrbl..... innerhalb der Tabelle gebrauchte, sonstige Variabel.

Übersicht

XML	RDF (Turtle)	Anmerkungen
<?xml version='1.0' encoding='UTF-8'?> <Reports xmlns="http://www.niso.org/schemas/counter" >	@prefix count: <http://www.platzhalter.de/counter#> . @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>. @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>. @prefix data: <http://www.platzhalter.de/erm/data#> .	Einbinden der Namespaces.
<Report	data:Rprt_x a count:Report .	
Created="*dtTm1*"	data:Rprt_x count:wasCreatedOn "*dtTm1*" .	Das Jahr von *dtTm1* (die ersten vier Zeichen) als vrbl1 speichern, später genutzt für Report-Label
ID="*strg1*"	data:Rprt_x count:hasReportID "*s1*" .	
Version="*strg2*"	data:Rprt_x count:hasReportVersion "*s2*" .	
Name="*strg3*"	data:Rprt_x count:hasReportTitle count:*strg3* .	
Title="*strg4*">	-	Standardisiert, daher bereits im Vokabular definiert und mit Instanzen von :ReportTitle verknüpft.
<Vendor>	data:Vndr_x a count:Vendor . data:Vndr_x count:creates data:Rprt_x.	
<Name> *strg5* </Name>	data:Vndr_x rdfs:label "*strg5* [Vendor]" . data:Rprt_x rdfs:label "vrbl1 strg3 *strg5*" . data:n1 a rdf:statement . data:n1 rdf:subject data:Vndr_x . data:n1 rdf:predicate hasOrganizationName . data:n1 rdf:object "*strg5*" . data:n1 count:sourceReport data:Rprt_x .	Bessere Lesbarkeit. Verwendung des Namens in diesen Labels ist kritisch, kann bei Veränderungen zu doppelten Labels führen. Wird zwar für URI-Vergabe genutzt, geänderter Name aber nicht gleichbedeutend mit geänderter Organsiation.. Zwei getrennt erstellte Ressourcen sollen daher manuell über owl:sameAs zusammengefügt werden. Dann sind durch die Gleichsetzung doch zwei zu unterscheidende Namen vorhanden, daher Reifikation.

<ID> *strg6* </ID>	data:n2 a rdf:statement . data:n2 rdf:subject data:Vndr_x . data:n2 rdf:predicate count:hasOrganizationID . data:n2 rdf:object "*strg6*" . data:n2 count:sourceReport data:Rprt_x .	Reifikation trotz Nutzung zur URI-Vergabe. Siehe oben.
<Contact>	data:Cntc_x a count:Contact . data:n3 a rdf:statement . data:n3 rdf:subject data:Vndr_x . data:n3 rdf:predicate count:hasContact . data:n3 rdf:object Cntc_x . data:n3 count:sourceReport data:Rprt_x .	Reifikation.
<Contact> *strg7* </Contact>	data:n4 a rdf:statement . data:n4 rdf:subject data:Cntc_x . data:n4 rdf:predicate count:hasContactName . data:n4 rdf:object "*strg7*" . data:n4 count:sourceReport data:Rprt_x .	Reifikation.
<E-mail> *strg8* </E-mail>	data:n5 a rdf:statement . data:n5 rdf:subject data:Cntc_x . data:n5 rdf:predicate count:hasEmail. data:n5 rdf:object <mailto:strg8> . data:n5 count:sourceReport data:Rprt_x .	Reifikation trotz Nutzung zur URI-Vergabe. Siehe oben.
</Contact>		
<WebSiteUrl> *aUri1* </WebSiteUrl>	data:n6 a rdf:statement . data:n6 rdf:subject data:Vndr_x . data:n6 rdf:predicate count:hasWebsiteURL . data:n6 rdf:object <*aUri1*> . data:n6 count:sourceReport data:Rprt_x .	Reifikation.

<LogoUrl> *aUri2* </LogoUrl>	data:n7 a rdf:statement . data:n7 rdf:subject data:Vndr_x . data:n7 rdf:predicate count:hasLogoURL . data:n7 rdf:object <*aUri2*> . data:n7 count:sourceReport data:Rprt_x .	Reifikation
</Vendor>		
<Customer>		
<Name> *strg9* </Name>	data:Cstm_x rdfs:label "*strg9* [Customer]" . data:n8 a rdf:statement . data:n8 rdf:subject data:Cstm_x . data:n8 rdf:predicate count:hasOrganizationName . data:n8 rdf:object "*strg9*" . data:n8 count:sourceReport data:Rprt_x .	Bessere Lesbarkeit. Verwendung des Namens im Label ist kritisch, kann bei Veränderungen zu doppelten Labels führen. Reifikation trotz Nutzung zur URI-Vergabe. Siehe oben.
<ID> *strg10* </ID>	data:Cstm _x count:hasOrganizationID "*strg10*" data:n9 a rdf:statement . data:n9 rdf:subject data:Cstm_x . data:n9 rdf:predicate count:hasOrganizationID. data:n9 rdf:object "*strg10*" . data:n9 count:sourceReport data:Rprt_x .	Reifikation trotz Nutzung zur URI-Vergabe. Siehe oben.
<Contact>	data:Cntc_y a count:Contact . data:n10 a rdf:statement . data:n10 rdf:subject data:Cstm_x . data:n10 rdf:predicate count:hasContact . data:n10 rdf:object Cntc_y . data:n10 count:sourceReport data:Rprt_x .	Reifikation.

<Contact> *strg11*</Contact	data:n11 a rdf:statement . data:n11 rdf:subject data:Cntc_y . data:n11 rdf:predicate count:hasContactName . data:n11 rdf:object "*strg11*" . data:n11 count:sourceReport data:Rprt_x .	Reifikation.
<E-mail> *strg12* </E-mail>	data:n12 a rdf:statement . data:n12 rdf:subject data:Cntc_y . data:n12 rdf:predicate count:hasEmail . data:n12 rdf:object <mailto:*strg12*>. data:n12 count:sourceReport data:Rprt_x .	Reifikation trotz Nutzung zur URI-Vergabe. Siehe oben.
</Contact>		
<WebSiteUrl> *aUri3* </WebSiteUrl>	data:n13 a rdf:statement . data:n13 rdf:subject data:Cstm_x . data:n13 rdf:predicate count:hasWebsiteURL . data:n13 rdf:object <*aUri3*> . data:n13 count:sourceReport data:Rprt_x .	Reifikation
<LogoUrl> *aUri4* </LogoUrl>	data:n14 a rdf:statement . data:n14 rdf:subject data:Cstm_x . data:n14 rdf:predicate count:hasLogoURL . data:n14 rdf:object <*aUri4*> . data:n14 count:sourceReport data:Rprt_x .	Reifikation
<Consortium>	data:Cnsr_x a count:Consortium . data:n15 a rdf:statement . data:n15 rdf:subject data:Cstm_x . data:n15 rdf:predicate count:isPartOfConsortium . data:n15 rdf:object Cnsr_x . data:n15 count:sourceReport data:Rprt_x .	Reifikation.

<Code> *strg13* </code>	data:n16 a rdf:statement . data:n16 rdf:subject data:Cnsr_x . data:n16 rdf:predicate count:hasConsortiumCode . data:n16 rdf:object "*strg13*" . data:n16 count:sourceReport data:Rprt_x .	Reifikation.
<WellKnownName> *strg14* </WellKnownName>	data:n17 a rdf:statement . data:n17 rdf:subject data:Cnsr_x . data:n17 rdf:predicate count:hasConsortiumName. data:n17 rdf:object "*strg14*" . data:n17 count:sourceReport data:Rprt_x .	Reifikation trotz Nutzung zur URI-Vergabe. Siehe oben.
</Consortium>		
<InstitutionalIdentifier> - </Institutionalidentifier>	-	Noch keine Nutzung laut COUNTER.
<ReportItems>	data:RpIt_x a count:ReportItem . data:RpIt_x count:isContainedIn data:Rprt_x .	
<ItemIdentifier>		
<Type>DOI</Type> *alt.* <Type>Proprietary</Type> *alt.* <Type>PrintISBN</Type> *alt.* <Type>PrintISSN</Type> *alt.* <Type>OnlineISBN</Type> *alt.* <Type>OnlineISSN</Type>		Mögliche Werte für das Type-Element, unten entsprechende Alternative auswählen. Die veralteten Type EISSN, ISSN sowie ISBN werden behandelt wie OnlineISSN, PrintISSN und PrintISBN.
<Value>*strg15*</Value>	data:RpIt_x hasDOI <http://doi.org/*strg15*> . *alt.* data:RpIt_x hasProprietaryID "*strg15*" . *alt.* data:RpIt_x hasPrintISBN <urn:ISBN:*strg15*> *alt.* data:RpIt_x hasPrintISSN <urn:ISSN:*strg15*> *alt.* data:RpIt_x hasOnlineISBN <urn:ISBN:*strg15*> *alt.* data:RpIt_x hasOnlineISSN <urn:ISSN:*strg15*>	Überprüfen, ob *strg5* das korrekte Format für den jeweiligen Identifier besitzt. Bei manchen Anbietern Angaben wie N/A im Value-Elementen
</ItemIdentifier>		

<ItemPlatform> *strg16* </ItemPlatform>	data:Pltf_x a count:Platform . data_Pltf_x rdfs:label "*strg16* [Platform]" . data:n18 a rdf:statement . data:n18 rdf:subject data:RpIt_x . data:n18 rdf:predicate count:hasPlatform . data:n18 rdf:object data:Pltf_x . data:n18 count:sourceReport data:Rprt_x .	Reifikation.
<ItemPublisher> *strg17* </ItemPublisher>	data:Pbls_x a count:Publisher . data_Pbls_x rdfs:label "*strg17* [Publisher]" . data:n19 a rdf:statement . data:n19 rdf:subject data:RpIt_x . data:n19 rdf:predicate count:hasPublisher . data:n19 rdf:object data:Pbls_x . data:n19 count:sourceReport data:Rprt_x .	Reifikation.
<ItemName> *strg18* </ItemName>	data:RpIt_x rdfs:label "*strg18* [ReportItem]" . data:n20 a rdf:statement . data:n20 rdf:subject data:RpIt_x . data:n20 rdf:predicate count:hasItemName . data:n20 rdf:object "*strg18*" . data:n20 count:sourceReport data:Rprt_x . *alt.* data:RpIt_x count:hasItemName "*strg18*" .	Bessere Lesbarkeit. Verwendung des Namens im Label ist kritisch, kann bei Veränderungen zu doppelten Labels führen. Reifikation. Wenn DOI, ISSN oder ISBN zur URI-Vergabe genutzt warden. Alternativ: Keine Reifikation. Wenn Titel zur URI-Vergabe genutzt wird.
<ItemDataType> *elmt1* </ItemDataType>	data:RpIt_x count:hasDatatype count:*elmt1* .	

<ItemPerformance		
PubYr="*year1*"> *alt.* PubYrFrom="*year2*" PubTo="*year3*">	data:DtRg_x a count:DateRange . data:DtRg_x rdfs:label "*year1*" . data:DtRg_x count:hasStartDay "*year1*-01-01" . data:DtRg_x count:hasEndDay "*year1*-12-31" . *alt.* data:DtRg_x rdfs:label "*year2*-*year3*" . data:DtRg_x count:hasStartDay "*year2*-01-01" . data:DtRg_x count:hasEndDay "*year3*-12-31" .	Angabe nur eines berücksichtigten Publikationsjahres. Berücksichtigung mehrerer Publikationsjahre.
<Period>	data:DtRg_y a count:DateRange .	
<Begin> *date1* </Begin>	data:DtRg_y count:hasStartDay "*date1*" . data:DtRg_y rdfs:label *vrbl2* .	Das Jahr und den Monat von *date1* (die ersten sieben Zeichen) als *vrbl2* speichern, genutzt für Label des Zeitraums.
<End> *date2* </End>	data:DtRg_y count:hasEndDay "*date2*" .	
</Period>		
<Category> *elmt2* </Category>	*s. u.*	Wird erst mit der Zählinstanz verknüpft.
<Instance>	data:CntI_x a count:CountingInstance . data:RpIt_x count:hasPerformance data:CntI_x . data:CntI_x count:considersPubYear data:DtRg_x . data:CntI_x count:measuredForPeriod data:DtRg_y . data:CntI_x count:hasCategory count:*elmt2* .	Der hierarchische XML-Aufbau wurde nicht in das RDF-Vokabular übernommen. Daher werden diese Daten direkt mit jeder Zählinstanz-Ressource verbunden, anstatt wie im XML in einer höheren Hierarchiestufe zusammengefasst zu werden. Dies vereinfacht SPARQL-Abfragen.
<MetricType> *elmt3* </MetricType>	data:CntI_x count:hasMetricType count:*elmt3* .	
<Count> *Intg1* </Count>	data:CntI_x count:hasCount "*Intg1*" .	
</Instance>		

<Instance>	data:CntI_y a count:CountingInstance . data:RpIt_x count:hasPerformance data:CntI_y . data:CntI_y count:considersPubYear data:DtRg_x . data:CntI_y count:measuredForPeriod data:DtRg_y . data:CntI_y count:hasCategory count:*elmt2* .	Siehe oben. Hier die Wiederholung der Daten für eine neue Zählinstanz.
<MetricType> *elmt4* </MetricType>	data:CntI_y count:hasMetricType count:*elmt4* .	
<Count> *Intg2* </Count>	data:CntI_y count:hasCount "*Intg2*" .	
</Instance>		
…	…	Restlichen, gleich strukturierten Inhalt wie oben behandeln.
</ItemPerformance>		
…	…	Restlichen, gleich strukturierten Inhalt wie oben behandeln.
</Customer>		
</Report>		
</Reports>		