Ontologien

Version 2008-04-27/0
Arbeitspaket AP1
verantwortlicher Partner: FH Worms

TextGrid

Modulare Plattform für verteilte und kooperative wissenschaftliche Textdatenverarbeitung - ein Community-Grid für die Geisteswissenschaften

Bundesministerium für Bildung und Forschung

Projekt: **TextGrid**

Teil des D-Grid Verbundes und der deutschen e-Science Initiative

BMBF Förderkennzeichen: 07TG01A-H

Laufzeit: Februar 2006 - Januar 2009

Dokumentstatus: final

Verfügbarkeit: öffentlich

Autoren:

C. Ludwig, FH Worms

M. W. Küster, FH Worms

1 Einleitung

Semantische Technologien haben spätestens seit dem Schlagwort *Web 2.0* , das u. a. semantisch angereicherte Informationen bieten soll, sehr viel Aufmerksamkeit erregt. Um Daten semantisch auswerten zu können, bedarf es grundsätzlich geeigneter Ontologien, weil nur in deren Kontext aus den Daten echte Informationen werden.

Semantische Anwendungen und speziell auch Ontologien sind für die TextGrid -Zielgruppe als Werkzeug und als eigenständiger Forschungsgegenstand gleichermaßen relevant. In diesem Report betrachten wir allerdings ausschließlich technische Lösungen zum Umgang mit Ontologien und für die Verwaltung von semantischen Informationen. Die Fragen, wie TextGrid die Arbeit an bzw. die Nutzung von Ontologien als textwissenschaftliche Objekte unterstützen kann, wurden bereits ausführlich in Report 5.1 [BÜ08] diskutiert.

Abschnitt 2 klärt die Verwendung der Begriffe *Ontologie*, *Referenzmodell* und *Referenzarchitektur* für die Zwecke dieses Reports und gibt anhand des Referenzmodells des Internationalen Komitees für Museumsdokumentation[1] ein konkretes Beispiel. Abschnitt 3 betrachtet UML, RDF sowie Topic Maps als Standards, die zur Beschreibung von Ontologien und zur Speicherung von Instanzdaten herangezogen werden können. Davon hängen die Möglichkeiten, in diesen Daten zu recherchieren ab, weshalb Abschnitt 4 die jeweils einschlägigen Abfragesprachen beschreibt. Weil es inhaltlich eng an dieses Thema angelehnt ist, stellt Abschnitt 5 – ein wenig abweichend von der strikten Aufgabenstellung von AP1, das in erster Linie existierende Techniken und Software zur Nutzung in TextGrid evaluiert – die in TextGrid entwickelte Lösung der Abbildung projektspezifischer Ontologien zur Textauszeichnung für den Zweck korpusübergreifender Recherchen vor. Jede Diskussion einer Recherche in Resourcebeschreibungen, denen eine gemeinsame Ontologie zu Grunde liegt, wäre unvollständig, wenn sie Registries außer Acht ließe. Deshalb stellt Abschnitt 6 die Resource Registry Standards UDDI und ebXML vor, zeigt Gründe für deren Scheitern in der Praxis auf und identifiziert Grundanforderungen an (föderierte) Registries.

2 Begriffsklärung, grundlegende Konzepte und Beispiele

2.1 Semantische Interoperabilität

Ein auf Dauer überlebensfähiges eHumanities-Ökosystem muss Dienste und Daten auch über die unmittelbaren Grenzen von TextGrid hinweg einbinden zu können. Andere Teilsysteme wie EPPT, TAPoR und Bricks, aber auch Daten aus Archiven, Verlagen und Museen müssen eingebunden und nachgenutzt werden. Umgekehrt ist es auch notwendig, TextGrids Dienste und Daten extern einsetzen zu können [KLA07].

Eine wesentliche Voraussetzung dafür ist neben der rein technischen die semantische Interoperabilität zwischen den verschiedenen Teilsystemen. Beschreibungen von Diensten und Daten müssen zwischen den Systemen austauschbar sein, um beide in einem nur locker verknüpften Ökosystem finden und nutzen zu können. Im Sinne des SOA Reference Model [ML+06] entspricht dies der Anforderung an die Sichtbarkeit (visibility) von Diensten sowie der (nicht notwendig vollständig maschinenlesbaren)

1 http://cidoc.mediahost.org/

Beschreibung von deren Zweck und Nutzungskonditionen (service descriptions und policies), die sich alle zusammen unter dem Begriff der Awareness zusammenfassen lassen.

Semantische Interoperabilität setzt allerdings nicht die gleichförmige Beschreibung entsprechender Objekte in identischen Formaten voraus. Eine solche Homogenität der Modelle und Formate wäre angesichts der Heterogenität der einzelnen Ansätze in den verschiedenen Diensten und Datensätzen in den meisten Bereichen kaum erreichbar. Schon innerhalb von TextGrid werden einzelne Projekte notwendigerweise ihre eigenen Encoding-Regeln für Texte einsetzen, welche den Spezifika der jeweiligen Textgattungen, Zeiten und Autoren Rechnung tragen. Voraussetzung für die Interoperabilität ist vielmehr die Möglichkeit, Teilmengen der jeweiligen Beschreibungen auf gemeinsame Modelle abzubilden, die es ermöglichen, zumindest wesentliche semantische Informationen über Projektgrenzen hinweg auszutauschen.

2.2 Ontologien

Wir machen uns für die Zwecke dieses Reports die gleiche Definition des Begriffs *Ontologie* zu eigen, die bereits im TextGrid-Report 5.1 „Ontologien und Wortnetze“ verwandt wurde: Eine Ontologie ist eine „formal explicit specification of a shared conceptualization for a domain of interest“ [GR93]. Ziel ist demnach, die Konzepte zu beschreiben, die den in einem Anwendungsgebiet verwendeten und ausgetauschten Informationen zu Grunde liegen.

Eine effektive Kommunikation ist ohne eine zumindest weitgehende Übereinkunft zwischen den Kommunikationspartnern bezüglich der verwendeten Konzepte schwer vorstellbar; Missverständnisse sind andernfalls geradezu vorprogrammiert. In einer Ontologie werden deshalb explizite Beschreibungen der Konzepte eines Fach- oder Anwendungsgebietes sowie der Beziehungen zwischen diesen Konzepten zusammengefasst. Durch Bezugnahme auf diese Ontologie kann dann klargestellt werden, wie bestimmte Informationen zu verstehen sind.

Insofern sind Ontologien keineswegs auf den Einsatz in der elektronischen Informationsverarbeitung beschränkt. Im Kontext von TextGrid stehen freilich Ontologien im Vordergrund, deren Beschreibungen maschinell auswertbar sind, so dass wir uns im Folgenden auf diesen Fall konzentrieren.

Der Begriff Ontologie wird häufig auch für Datensammlungen benutzt, die korrekter als *Wissensdatenbanken* (engl. *knowledge bases)* bezeichnet werden müssten. Ontologien beschränken sich auf Beschreibungen der Konzepte, wohingegen Wissensdatenbanken auch Fakten und Aussagen enthalten, die den aktuellen Stand eines Fachgebietes beschreiben. Eine Wissensdatenbank muss sich dafür auf eine Ontologie beziehen, ihr eigentlicher Gegenstand ist aber das Faktenwissen. Der Unterschied lässt sich an einem Beispiel aus der Biochemie verdeutlichen: Eine Protein-Ontologie wird typischerweise das Konzept „physiologische Funktion“ definieren, aber keine Beschreibungen konkreter Proteine enthalten. Eine einschlägige Wissensdatenbank wird hingegen zu jedem Protein alle bekannten physiologischen Funktionen aufführen und für das verwendete Konzept (explizit oder implizit) auf die Ontologie verweisen. Die Wissensdatenbank wird ständig wachsen und auch zu bekannten Proteinen werden u. U. neu entdeckte physiologische Funktionen hinzukommen. Die Ontologie sollte hingegen stabil bleiben. Diese begriffliche Verwirrung wird leider dadurch verstärkt, dass in rdf+xml serialisierte Wissensdatenbanken, die auf eine in OWL (s. u.) spezifizierte Ontologie bezogene Instanzdaten enthalten, regelmäßig mit <ontology>-Elementen eingeleitet werden, auch

wenn die OWL Spezifikation ausdrücklich darauf hinweist, dass die Dokumente deshalb nicht notwendig Ontologien beschreiben.

2.3 Referenzmodelle und Referenzarchitekturen

Ein Referenzmodell ist in den Worten von OASIS ([OS08], Hervorhebung hinzugefügt)

> an abstract framework for understanding significant relationships among the entities of some environment, and for the development of consistent standards or specifications supporting that environment. A reference model is based on a small number of unifying concepts [...] A reference model is not directly tied to any standards, technologies or other concrete implementation details, but it does seek to *provide a common semantics that can be used unambiguously across and between different implementations*.

Seiner Natur nach betont ein Referenzmodell die Semantik der in einer Domain verwendeten Kernkonzepte und -beziehungen. Referenzarchitekturen bilden die abstrakten Konzepte und Beziehungen der Referenzarchitektur auf immer noch abstrakte Modelle in definierteren Anwendungsszenarien ab [ML+06, p. 4], die dann in konkreten Architekturen und schlussendlich in Implementierungen ihren Niederschlag finden.

Für Ontologien verwendet man für die Darstellung von Referenzmodellen und Referenzarchitekturen, teils auch von konkreten Architekturen, gerne (u. U. erweiterte) UML-Klassendiagramme. Diese Architekturen können dann in technischen Standards wie RDF+OWL oder Topic Maps implementiert werden, vgl. Abschnitt 3.

Das Verhältnis zwischen Refernzmodellen und konkreten Realisierungen soll im nächsten Abschnitt durch zwei Beispiele illustriert werden.

2.4 Beispiel: CIDOC CRM

Das *CIDOC Conceptual Reference Model (CRM)* ist in seinen eigenen Worten „a formal ontology intended to facilitate the integration, mediation and interchange of heterogeneous cultural heritage information“ [ISO06, 4.2 Introduction] mit dem Ziel, „semantic definitions and clarifications needed to transform disparate, localised information sources into a coherent global resource, be it within a larger institution, in intranets or on the Internet“ [ISO06, Objectives] bereitzustellen. In diesem Sinne ist CIDOC-CRM nicht nur dem Namen nach ein typisches Referenzmodell für Informationen aus dem Cultural Heritage Sektor, zu dem auch wesentliche Metadaten der in TextGrid und anderen eHumanities-Systemen vorhandenen Ressourcen gehören.

CIDOC-CRM definiert eine Klassenhierarchie von 81 (notwendigerweise größtenteils immer noch recht abstrakten) Klassen und 132 verschiedenen Eigenschaften, die auf höchster Ebene auf die Oberklassen Temporal Entity (E2), Persistent Item (E77), Time Span (E52), Place (E53) und Dimension (E54) zurückgeführt sind, die ihrerseits Unterklassen von CRM Entity (E1) sind. Die Semantik jeder Klasse und jeder Eigenschaft wird im Detail spezifiziert.

CIDOC-CRM kann in verschiedensten Formaten ausgedrückt werden. Das im Standard selbst verwendete Format OWL ist ausdrücklich nur ein Format unter vielen möglichen.

Wie jedes Referenzmodell soll CIDOC-CRM nicht *as is* eingesetzt werden. Es muss durch Ableitungen von den vorhandenen abstrakten Konzepten konkretisiert werden, die dann entweder

direkt bearbeitet oder (wesentlich häufiger) aus bestehenden Formaten erzeugt werden können, wobei diese Abbildungen üblicherweise verlustbehaftet sind, da konkrete Formate lokale Anforderungen und Besonderheiten berücksichtigen können und sollen (s. „compatibility with the CRM“). In beiden Fällen ist es durch die definierte Semantik der Konzepte und Beziehungen im CIDOC-CRM möglich, über die Datensätze zu schlussfolgern (*reasoning*).

Ein ganz wesentlicher Teil der Arbeit im Kontext des Standards sind Abbildungsregeln zwischen anderen, im Bereich der *Cultural Heritage Documentation* eingesetzten Daten- und Austauschformaten wie Dublin Core, dem Austauschformat der *Art Museum Image Consortium (AMICO)*, der *Encoded Archival Description (EAD)*, FRBR (*International Federation of Library Associations and Institutions Functional Requirements for Bibliographic Records*) usw., aber auch den konkreten Formaten, wie sie in zahlreichen nationalen und regionalen Museen eingesetzt werden[2].

3 Standards zur Ontologiebeschreibung

Es gibt kein ubiquitäres Format zur Beschreibung von Ontologien. Ausgehend von verschiedenen Anforderungen haben sich diverse Beschreibungssprachen herausgebildet, so dass für jeden Einsatzzweck die jeweils am besten geeignete gewählt werden muss. Wir werden im Folgenden UML, RDF mit OWL sowie Topic Maps genauer betrachten.

3.1 UML

Als Anfang der 1990er Jahre das objektorientierte Programmierparadigma starke Verbreitung fand, entstand ein Bedarf an Werkzeugen, die die objektorientierte Modellierung von Softwaresystemen visuell unterstützen. Nachdem zunächst proprietäre Formate dominierten, wurde 1997 die Spezifikation der *Unified Modelling Language (UML)* von dem Industriekonsortium *Object Management Group* akzeptiert. UML hat sich seither auf dem Markt durchgesetzt und wurde inzwischen auch bei der ISO/IEC formal standardisiert [ISO05].

Die im objektorientierten Design modellierten Klassen sind regelmäßig nichts anderes als Ausformulierungen von Konzepten und Spezifikationen der Eigenschaften oder Attribute, die mit konkreten Instanzen dieser Konzepte verbunden sind. Durch Vererbung entstehen Klassenhierarchien, die Ausdifferenzierungen der jeweiligen Konzepte beschreiben. Es lag deshalb von Anfang an nahe, UML auch für die Beschreibung von Ontologien heranzuziehen.

Betrachtet man die Entstehungsgeschichte, so wird deutlich, dass UML zunächst nicht für diesen Einsatzzweck entworfen wurde. Dies ist der Grund dafür, dass sich manche für Ontologien typische Phänomene in UML nur schwer oder gar nicht ausdrücken lassen. So gibt es in UML wie in Ontologien Klassen / Instanz-Beziehungen; Objekte – also Instanzen einer Klasse – sind in UML grundsätzlich verschieden von Klassen und können nicht deren Rolle übernehmen, während dies in Ontologien durchaus möglich ist, wenn darin mehr als zwei Abstraktionsebenen modelliert sind. Auch für die Beschreibung der Eigenschaften von Konzepten ist UML im Allgemeinen nicht ausdrucksstark genug; in Ontologien ist es ohne weiteres möglich, dass eine Eigenschaft eine Verallgemeinerung anderer Eigenschaften ist, was in UML nicht vorgesehen ist.

2 http://cidoc.ics.forth.gr/scope.html

Dennoch kann eine Ontologie-Modellierung in UML in vielen Fällen möglich und sinnvoll sein; insbesondere die Visualisierung von Ontologien durch statische UML-Diagramme ist regelmäßig hilfreich. Zudem wird die Unterstützung für Ontologien in UML nach und nach verbessert [OMG07] und es lassen sich durch geeignete Konventionen häufig hinreichende Lösungen finden, z. B. durch Verwendung von UML *Stereotypes*.

3.2 RDF + OWL

Das Resource Description Framework

Das vom W3C als Recommendation standardisierte *Resource Description Framework (RDF)* [W3C04] entstand primär, um Ressourcen im World Wide Web beschreiben zu können. Aufgrund der konsequenten Verwendung von URIs können mit RDF auch Aussagen über Ressourcen ausgedrückt werden, die nicht selbst im Netz verfügbar sind – also sowohl materielle Dinge, die übers Web vielleicht bestellt, aber nicht heruntergeladen werden können, als auch abstrakte Konzepte. Bei der Entwicklung von RDF stand immer die maschinelle Auswertbarkeit der Aussagen im Vordergrund.

Aus diesem Grund sind die einzelnen Aussagen in RDF sehr einfach und immer gleich aufgebaut: es handelt sich immer um ein Tripel aus *Subjekt, Prädikat* (häufig auch *Property* genannt) und *Objekt.* Jedes Element eines solchen Tripels wird durch einen URI (oder genauer: eine URI-Referenz) identifiziert; im Fall der Objekte werden auch Literale (Strings, Datumsangaben etc.) unterstützt.

Aufgrund der Einschränkung auf Tripel können in RDF nur binäre Relationen unmittelbar ausgedrückt werden. Für komplexere Relationen müssen zusätzliche Relationen und (in der Regel unbenannte) Zwischenressourcen eingefügt werden. Um eine scheinbar einfache Aussage in einem RDF-Dokument zu lesen, kann es deshalb notwendig sein, einen größeren RDF-Teilgraphen zu parsen. Dies stellt bei der maschinellen Verarbeitung üblicherweise kein Problem dar, macht aber die enthaltenen Informationen für menschliche Leser deutlich schwerer greifbar.

RDF-Tripel sind selbst keine adressierbaren RDF-Objekte. Gelegentlich ist es aber notwendig, Aussagen zu qualifizieren und somit Aussagen über Aussagen (also RDF-Tripel) zu machen. Beispielsweise kann es im Rahmen der Dokumentation der Qualitätssicherung in einer Firma erforderlich sein, folgende Aussagen festzuhalten: „Person X hat zugesagt, dass Bauteil Y die Last Z tragen kann.“ Der hintere Teil dieser Aussage lässt sich leicht als Tripel ausdrücken: (Bauteil Y, hat-maximale-Last, Last Z). Allerdings sieht RDF (mit Ausnahme der Literale) ausschließlich *Dinge* als Subjekte oder Objekte vor, die über eine URI-Referenz identifiziert werden können. Wir haben also zunächst keine Möglichkeit, ein Tripel (Person X, hat-gesagt, ξ) zu definieren, in dem ξ für das zuvor definierte Tripel über Y und Z steht. ξ muss zuvor „verdinglicht“ oder *reifiziert* werden. Dies geschieht, indem eine neue Ressource A für die zu reifizierende Aussage eingeführt wird, für die eine URI-Referenz angegeben werden kann. Das ursprüngliche Tripel ξ wird dann durch vier Tripel jeweils mit Subjekt A ersetzt:

1. (A, rdf:type, rdf:Statemenet)
2. (A, rdf:subject, Bauteil Y)
3. (A, rdf:predicate, hat-maximale-Last)
4. (A, rdf:object, Last Z)

Die zu Grunde liegende, qualifizierte Aussage kann offensichtlich rekonstruiert werden, ist aber wiederum für einen menschlichen Leser deutlich schwerer zu erfassen als zuvor. Die im oben angeführten Beispiel gewünschte Qualifikation erfolgt dann durch das zusätzliche Tripel (Person X, hat-gesagt, A).

RDF Schema

Bis dahin sind RDF-Dokumente nur Sammlungen von Aussagen über Objekte; durch die in *RDF Schema* – einem Teil der RDF-Spezifikation – definierten Properties und Ressourcen ist es auch möglich, Ressourcen als Konzepte zu deklarieren, Ober- und Unterklassen- bzw. Instanzbeziehungen zu modellieren oder Properties auf bestimmte zulässige Subjekt- und Objekttypen einzuschränken.

Dies erlaubt, eigene Vokabulare zu definieren; so ist z. B. das Dublin Core Vokabular vollständig als RDF Schema beschrieben. Für viele Ontologien ist RDF Schema als Beschreibungssprache hinreichend mächtig, insbesondere dann, wenn auf zusätzliche Informationen und Randbedingungen, die für die automatische Schlussfolgerung mitunter relevant sind, verzichtet werden kann.

Die Web Ontology Language

RDF kann allerdings erst in Verbindung mit der ebenfalls als W3C Recommendation standardisierten *Web Ontology Language (OWL)* [W3C04a] zur Beschreibung allgemeiner Ontologien eingesetzt werden, weil OWL ermöglicht, das verwendete Vokabular mitsamt Randbedingungen und Abhängigkeiten sehr viel detaillierter zu spezifizieren, etwa durch Angabe von Kardinalitäten, zusätzlicher Charakteristika von Properties oder Relationen zwischen Klassen bzw. Properties. In OWL kann beispielsweise ausgedrückt werden, dass Instanzen einer Klasse immer über bestimmte Properties verfügen müssen.

Syntaktisch ist OWL eine Erweiterung des RDF-Vokabulars, wie sie in RDF bereits vorgesehen ist. In OWL formulierte Ontologie-Spezifikationen sind also valide RDF-Dokumente.

Ähnlich wie bei RDF stand auch bei der Entwicklung von OWL die maschinelle Auswertbarkeit im Vordergrund, insbesondere mit dem Ziel, automatisiertes Schlussfolgern zu ermöglichen. Deshalb definiert die OWL-Spezifikation auch zwei echte Teilsprachen OWL-Lite und OWL-DL, von denen nachgewiesen ist, dass jede aus den darin ausgedrückten Aussagen mögliche Schlussfolgerung auch berechenbar ist. Im Gegenzug kann beispielsweise eine Ressource nicht gleichzeitig Klasse und selbst Instanz einer anderen Klasse sein. Dies ist bei Nutzung des vollen Sprachumfangs von OWL durchaus möglich, allerdings auf Kosten der Berechenbarkeitsgarantien.

Die Betonung der maschinellen Auswertbarkeit spiegelt sich z.B auch in den Charakteristiken wider, die Properties zugewiesen werden können: Durch diese kann festgelegt werden, dass Properties etwa symmetrische oder transitive Relationen oder Funktionen wiedergeben. Oder für bestimmte Instanzen (*Individuen* in der OWL-Terminologie) kann deklariert werden, dass sie als äquivalent oder verschieden betrachtet werden sollen. Bei Verwendung solcher Deklarationen kann es allerdings vorkommen, dass die von einem oder mehreren Dokumenten beschriebene Wissensdatenbank nicht mehr widerspruchsfrei ist.

3.3 Topic Maps

Grundlagen

Topic Maps ist eine Famile von ISO/IEC-Standards – teils noch im *Draft*-Status – [ISO03, ISO06a, ISO07, DNB07, GB07, MBN05] zur strukturierten Beschreibung von Informationen über Ressourcen und den dazwischen bestehenden Beziehungen, vgl. Abb. 1. Insofern hat dieser Standard ein vergleichbares Anwendungsgebiet wie das im vorigen Abschnitt beschriebene RDF+OWL.

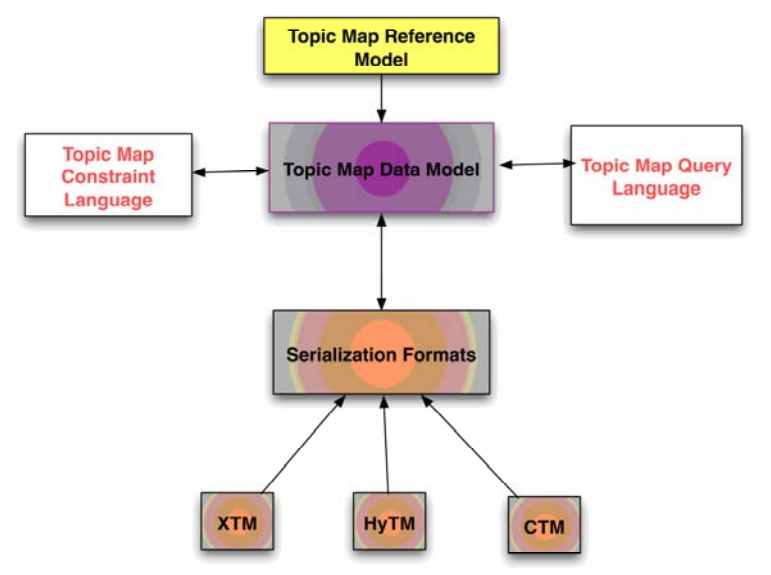

Abb. 1: Topic Maps Standards

Einem Artikel von Steve Pepper [PE00] folgend beschreiben wir Topic Maps anhand der grundlegenden Konzepte *Topic*, *Association*, *Occurrence* und *Scope*.

Topic: Jedes „Ding“, das mit einem Namen versehen werden kann, ist als Topic in einer Topic Map repräsentierbar. Dabei spielt es keine Rolle, ob es sich um eine elektronische Ressource, die über einen URL adressiert werden kann, ein materielles Objekt oder ein Abstraktum handelt. Innerhalb einer Topic Map trägt jedes Topic eine eindeutige ID; außerdem kann jedem Topic eine beliebige Zahl an URIs zugewiesen werden, wobei diese URIs als persistent angenommen werden.

Die Definition von Topics ist so allgemein gehalten, dass sowohl abstrakte Konzepte als auch Instanzdaten dadurch repräsentiert werden können. Dadurch entstehen verschiedene Typen von Topics, die ihrerseits wieder als Topics modelliert werden.

Topics können – abgesehen von den rein technischen IDs und URIs – beliebig viele Namen tragen. Denn was im Deutschen z. B. als „Stuhl“ bezeichnet wird, heißt im Englischen „chair“.Diese Namen sind immer an einen Kontext – den *Scope*, s. u. - gebunden. Denn etwa der Name „Hut“ hat eine sehr verschiedene Bedeutung, je nachdem ob er in einem deutschen oder englischen Kontext interpretiert wird – mal als Kopfbedeckung, mal als Hütte.

Association: Eine Sammlung von Topics ist für sich genommen nur selten hilfreich; der Nutzwert entsteht durch die Beschreibung der Beziehungen zwischen den Topics in Form der Associations. Topic Maps unterstützt nicht nur binäre, sondern n-äre Beziehungen. In einer Topic Map wird man in aller Regel die Associations wieder ihrem Typ nach gruppieren können, also alle Beziehungen des Typs „Autor-Werk“ oder „Binding für X nach Y definiert durch Z“. Jedem Teilnehmer in einer Beziehung kann obendrein explizit eine Rolle zugewiesen werden, falls die Beziehung nicht symmetrisch ist. Die Typen der Associations und die Rollen werden als Topics repräsentiert und referenziert. Die Reifikation einer konkreten Association erfolgt ebenfalls durch ein geeignetes Topic.

Auch Beziehungen sind kontextabhängig und können deshalb durch einen Scope qualifiziert werden: In einer Topic Map, die Hierarchien am Arbeitsplatz beschreibt, wird es eine Beziehung „ist

Vorgesetzter von“ geben. Diese Beziehung trifft im Einzelfall aber immer nur im Kontext eines konkreten Beschäftigungsverhältnisses zu; falls der Arbeitnehmer noch ein zweites Beschäftigungsverhältnis hat, spielt der Vorgesetzte am ersten Arbeitsplatz keine Rolle am zweiten.

Occurrence: In einer Topic Map werden abstrakte Konzepte ebenso wie konkrete Ressourcen durch Topics modelliert. Es gibt aber regelmäßig bereits existierende, addressierbare oder elektronisch kopierbare Ressourcen, die ein gegebenes Topic in irgendeiner Form beschreiben. Für ein Topic, das eine konkrete Person repräsentiert, könnte dies etwa ein Foto oder eine Biographie sein. Sofern in der Topic Map keine weitergehenden Aussagen über diese Ressourcen getroffen werden, ist es nicht unbedingt notwendig, dafür jeweils eigene Topics einzuführen und mit geeigneten Associations zu verknüpfen; statt dessen können sie als so genannte *Occurrences* an das ursprüngliche Topic gebunden werden.

Je nach der Form der Beschreibung lassen sich Occurrences in Typklassen einteilen – im vorstehenden Beispiel etwa in „Fotografie“ oder „Biographie“. Dieser Typ kann in der Topic Map als *Occurrence Role* festgehalten werden, wobei es sich um einen Verweis auf ein Topic handelt, das den jeweiligen Typ repräsentiert. Durch diesen Verweis wird die Occurrence als Instanz dies Typs ausgewiesen.

Beschreibungen eines Topics sind im Allgemeinen ebenfalls kontextabhängig. Der Kontext, in dem ein Occurrence zutreffend ist, kann wiederum durch einen geeigneten Scope expliziert werden.

Scope: Wir haben schon mehrfach erwähnt, dass Restriktionen auf einen gegebenen Kontext durch die Angabe eines geeigneten *Scopes* modelliert werden. Ein Scope wird durch die Angabe einer beliebigen Anzahl von Topics, den so genannten *Themes,* definiert. Der Scope ohne jedes Theme ist der *unconstrained scope*, der den Kontext aller Topics beschreibt. Andernfalls ist der Kontext, in dem die jeweilige Association, das Occurrence oder der Name betrachtet werden soll, der Schnitt aller aufgeführten Themes.

Topic Maps Constraint Language

Bis jetzt können wir in einer Topic Map sowohl Konzepte als auch Instanzdaten beschreiben und Beziehungen zwischen den einzelnen Elementen der Topic Map ausdrücken. Von Trivialbeispielen abgesehen wird eine Ontologie aber auch vorschreiben, über welche Occurrences und Associations Instanzdaten mindestens verfügen müssen, von Instanzen welchen Typs bestimmte Rollen in Beziehungen ausschließlich ausgefüllt werden können usw.

Derartige Randbedingungen können mit Hilfe der Topic Maps Constraint Language (TMCL) [MBN05] formuliert werden, deren Spezifikation derzeit als Arbeitsentwurf vorliegt. TMCL erlaubt, eine Sammlung von Instanzdaten bezüglich ihrer Konformität im Blick auf die jeweilige Ontologie zu validieren.

3.4 Interoperabilität zwischen RDF+OWL und Topic Maps

RDF + OWL und Topic Maps + TMCL sind im Wesentlichen gleich mächtig. Dies wird auch durch die Arbeit einer W3C Arbeitsgruppe unterstrichen, die Richtlinien [PPGU06] entwickelt, wie Ontologien und Instanzdaten aus dem einen in das andere Format überführt werden sollen. Die wichtigsten Unterschiede sind:

1. RDF+OWL kennt ausschließlich binäre Relationen; n-äre Relationen müssen durch binäre mit zusätzlichen (unbenannten) Ressourcen nachgebaut werden, wobei die zusätzlichen Ressourcen nicht notwendig eine natürliche Entsprechung in der modellierten Welt haben.
2. Der in Topic Maps vorhandene Scope macht es einfacher, verschiedene – potenziell widersprüchliche – „Weltsichten“ in einer Topic Map zusammenzufassen. Dies ist auch in RDF+OWL grundsätzlich möglich, allerdings wieder nur durch Einführung zusätzlicher Indirektionen, welche die Ontologie komplexer machen.
3. Weil in RDF ausnahmslos alles durch Tripel aus Subjekt, Prädikat und Objekt ausgedrückt wird, lassen sich RDF-Datenbanken („Triplestores“) vergleichsweise direkt implementieren. Das Datenmodell von Topic Maps ist wesentlich komplexer.

Zusammenfassend lässt sich sagen, dass immer dann, wenn eine maschinelle Auswertung der Daten im Vordergrund steht, RDF + OWL der Vorzug gegeben werden sollte; in vielen Fällen wird man sogar auf OWL verzichten können und mit RDFS auskommen.

Falls man jedoch erwartet, dass die zu erfassenden Daten sich nicht besonders gut für eine maschinelle Auswertung eignen, weil sie z. B. nicht hinreichend strukturiert sind oder etwa widersprüchliche Informationen absehbar sind, so sind Topic Maps vorteilhaft. Denn menschliche Endanwender können Ihre Suche anhand des durch den Scope beschriebenen Kontexts bewusst anpassen und so in komplexen Situationen häufig effizienter navigieren.

Auch immer dann, wenn die Instanzdaten durch Endanwender bereitgestellt werden sollen, gilt zu beachten, dass das Schema möglichst simpel gehalten werden sollte. Denn die Anwender tun sich leichter, wenn sie wenige, klar voneinander abgegrenzte Eigenschaften bzw. Relationen der Instanzen beschreiben sollen und für spezifische Details im Zweifel Freitextfelder nutzen können. Andernfalls muss damit gerechnet werden, dass die Bereitschaft zur Dateneingabe sinkt und die Qualität der bereitgestellten Informationen in den zahlreichen, nur in Nuancen unterschiedlichen Felder unbefriedigend ist. In solchen Situationen kommt also die Stärke von RDF +OWL nicht zum Tragen,

4 Abfragesprachen

Der Aufwand für die Definition einer Ontology und den Aufbau einer Knowledge Base wäre nicht gerechtfertigt, könnte man in den Daten nicht effiziente und komplexe Suchen ausführen. Weil es sowohl für RDF als auch für Topic Maps XML-Serialisierungsformate gibt, ist es grundsätzlich möglich, die Datenbestände z. B. mit Xpath-Ausdrücken zu durchsuchen; jedoch ist dies umständlich und mit viel „manuellem“ Aufwand zum Verknüpfen von Ergebnismengen verbunden, also im Allgemeinen nicht effizient. Deshalb existieren für beide Technologien spezialisierte Abfragesprachen, SPARQL und TMQL.

4.1 SPARQL

Die SPARQL Query Language for RDF wurde im Januar 2008 als Empfehlung des W3C [W3C08] standardisiert. Eine SPARQL-Abfrage definiert typischerweise zunächst diverse Präfixe, um die URIs der referenzierten Ressourcen und Relationen abkürzen zu können, gefolgt von einer von vier möglichen *Query* Forms (*SELECT*, *CONSTRUCT*, *DESCRIBE* und *ASK*). Die Query Form bestimmt, wie das Ergebnis der Suche ausgegeben wird: Als Liste, als neuer RDF-Graph oder einfach als boolescher Wert, der angibt, ob ein auf die angegebenen Bedingungen passender Eintrag gefunden wurde.

Die Bedingungen werden im Wesentlichen wieder als die von RDF bekannten Tripel formuliert, wobei einzelne Elemente der Tripel an Variablen gebunden werden können. Die Treffermengen können per Schnitt oder Vereinigung verknüpft werden und mittels *OPTIONAL* können bedingte Suchmuster realisiert werden.

SPARQL führt keine automatischen Inferenzen durch, etwa transitive Abschlüsse von Relationen etc., wie sie mittels OWL spezifiziert werden können. In solchen Fällen muss der Anwender den RDF-Graphen selbst mit geeigneten SPARQL-Abfragen durchlaufen.

4.2 TMQL

Die *Topic Maps Query Language* wird derzeit von ISO/IEC JTC1 SC34 WG3 standardisiert und soll zur ISO-Norm 18048 werden. Derzeit ist ein Working Draft [GB07] des Standards verfügbar.

Grundsätzlich besteht eine TMQL-Abfrage aus drei Teilen: Einer oder mehreren Bedingungen, durch die Elemente einer Topic Map herausgefiltert werden, von diesen ausgehend eine Navigation in der Topic Map entlang verschiedener Achsen, sowie schließlich Anweisungen für die Ausgabe der so gefundenen Daten. Als Achsen stehen Unter-/Oberklassenbeziehungen, Klassen-/Instanzbeziehungen, Scope, Occurrence u.a. zur Verfügung, so dass mit TMQL auf die gesamten Abstraktionselemente von Topic Maps zugegriffen werden kann.

Für die Formulierung von TMQL-Abfragen kann der Anwender zwischen drei im Wesentlichen gleich mächtigen Syntaxvarianten wählen. Eine an SQL angelehnte der Form `select` *`Ausgabespezifikation`* `where` *`Bedingungen`*, eine als *FLWR* bezeichnete Form, bei der über alle Elemente einer Menge iteriert wird und die gewisse Ähnlichkeiten mit XQuery hat, sowie Xpath-ähnlichen Pfadausdrücken. Die drei Formen können bzw. müssen zum Teil auch vermischt werden. Zwar ist je nach Anwendungsfall mal das eine, mal das andere Format bequemer zu nutzen; dennoch hinterlässt dieses Gemenge aus drei Abfrageformaten den etwas schalen Eindruck, dass es sich bei dem Standardentwurf um ein Kompromissdokument handelt, das existierende proprietäre Abfragesprachen unter einen Hut zu bringen versucht.

Mit Ausnahme der Ober-/Unterklassen- sowie der Klassen-/Instanzrelation verzichtet auch TMQL bewusst auf jede automatische Inferenz; nur bei den letztgenannten Beziehungen werden innerhalb der Klassenhierarchie auch indirekte Oberklassen berücksichtigt.

5 Recherche in TextGrid

Texte, die von einem Textwissenschaftler in TextGrid ausgezeichnet wurden, enthalten typischerweise zahlreiche Informationen, die über den Volltext und die strikten Metadaten hinausgehen. Die

Erschließung dieser Informationen für die Recherche in TextGrid ist ein offensichtliches Desiderat. Allerdings hat TextGrid bewusst davon abgesehen, ein für alle in TextGrid erstellten Projekte einheitliches Auszeichnungsschema zu entwerfen; angesichts der Unterschiede in den einzelnen Textgattungen, der spezifischen Fragestellungen in den einzelnen Projekten und dem in die Auszeichnung investierten Aufwand wäre es unrealistisch, dass ein fixiertes Auszeichnungsformat bei den Anwendern auf Akzeptanz stieße. Nicht ohne Grund lassen die Richtlinien der Text Encoding Initiative [BB07] dem Nutzer sehr viele Freiheiten, welche Metadaten erfasst und welche Auszeichnungselemente im einzelnen Projekt verwendet werden.

Um wie gewünscht die vom Editor bereitgestellten Daten projektübergreifend für die Recherche nutzen zu können, benötigt das Recherchetool allerdings Informationen darüber, welche Konzepte sich hinter welchen Auszeichnungen verbergen. Um dies trotz der potentiellen Vielzahl an spezifischen Auszeichnungsformaten realisieren zu können, hat TextGrid eine Reihe von *Base Line Encoding Schemas* (auch *Kernkodierungen* genannt) für die wichtigsten Textgattungen entwickelt. Es handelt sich dabei um auf TEI basierende XML-Instanziierungen von Ontologien, die in Editionen für die einzelnen Textgattungen typischerweise erfasste Phänomene beschreiben.

Einem Editionsprojekt steht es frei, wie es seine Texte in TextGrid auszeichnet. Das projektspezifische Schema muss sich nicht an das jeweilige Base Line Encoding Schema anlehnen; TextGrid verlangt nicht einmal zwingend die Verwendung von TEI. Statt dessen muss jedes Projekt einen *Adaptor* definieren, der Daten im projektspezifischen Format auf das Base Line Encoding abbildet. Bei der Realisierung der Adaptoren haben die Anwender weitgehende Freiheiten; typischerweise wird es sich aber um geeignete XSLT-Skripte handeln. Im Allgemeinen wird es nicht möglich sein, alle Informationen, die im projektspezifischen Auszeichnungsformat kodiert sind, im Base Line Encoding auszudrücken; die vom Adaptor ausgeführte Transformation wird also verlustbehaftet sein. Umgekehrt gibt es keine Garantie, dass die Projekte alle Informationen, die im Base Line Encoding festgehalten werden können, von jedem Projekt tatsächlich erfasst werden. Die Adaptoren implementieren folglich (im Allgemeinen) weder injektive noch surjektive Abbildungen zwischen den projektspezifischen Ontologien und den von TextGrid durch die Kernkodierungen vorgegebenen Ontologien.

Wenn ein Projekt Daten in TextGrid speichert, für die ein Adaptor definiert ist, dann werden diese nicht nur in ihrer Originalform, sondern zusätzlich in die jeweilige Kernkodierung transformiert und in einer XML-Datenbank hinterlegt. Hierdurch werden korpusübergreifende Recherchen ermöglicht, weil das Recherchetool hierfür auf die in Kernkodierung hinterlegten Daten (mit definierter Struktur und Semantik) zurückgreifen kann. Zugleich sind dem Anwender, der das in einem konkreten Projekt genutzte Auszeichnungsschema kennt, projektspezifische Recherchen zu Details nicht verbaut, die im Base Line Encoding nicht mehr enthalten sind.

6 Resource Registries[3]

Falls Ressourcen – seien es Daten oder Dienste – in einem Netzwerk zur Nutzung durch Dritte bereitgestellt werden sollen, so genügt es nicht, diese auf einen Webserver bzw. einen Webservice-Container hochzuladen; die potentiellen Anwender müssen auch in der Lage sein, die Ressourcen zu finden. Für Webseiten übernehmen regelmäßig Suchmaschinen wie Google diese Aufgabe, ohne dass

3 Dieser Abschnitt nutzt Material aus [KML07].

der Ressourcenprovider selbst aktiv werden müsste. (Wenn der *Page Rank* eine wesentliche Rolle spielt, etwa im kommerziellen Umfeld, dann muss der Ressourcenprovider freilich oft einen erheblichen Aufwand betreiben.) Doch abgesehen davon, dass die gängigen Suchmaschinen keine Suche anhand semantischer Beschreibung der Webseiten anbieten, helfen sie auch bei der Suche nach Webservices nicht weiter. Dies ist das Einsatzgebiet für eine *Registry,* also ein „structured repository for concepts with references to data objects“ [CEN06].

6.1 Existierende Standards für Resource Registries

Mit UDDI und ebXML existieren zwei Standards für Registries, die wir im Folgenden kurz vorstellen. Die Gründe, weshalb beide Standards in der Praxis nur selten zum Einsatz kommen, diskutieren wir im nächsten Abschnitt.

UDDI

Universal Description, Discovery and Integration (UDDI) ist sicher der bekannteste Standard für Service Registries, entwickelt unter der Federführung von IBM, Microsoft und Ariba und veröffentlicht als OASIS Standard [CHRR04]. Er wurde neben SOAP und WSDL als eine der Säulen Service-orientierter Architekturen betrachtet, die mittels Web Services implementiert wurden.

Technisch gesehen ist UDDI in erster Linie eine Schnittstelle für eine Sammlung SOAP-basierter Web Services zusammen mit den zugehörigen Datenmodellen. Diese Schnittstelle ist von Version zu Version des UDDI-Standards gewachsen und deckt mittlerweile neben anderem Methoden für die Veröffentlichung von Informationen über Firmen und von ihnen angebotenen Dienste, für das Auffinden dieser Informationen sowie für das Erstellen von Querverweisen zwischen ihnen ab.

UDDI sieht keine Kooperationsmodelle für Registries vor; lediglich Mechanismen zur Replikation werden in der Spezifikation angegeben.

Zunächst war UDDI eng verknüpft mit dem Konzept offener und freier, Web-basierter Registries, den so genannten *UDDI Operator Sites*, die Informationen über Firmen und andere Organisationen veröffentlichen sollten. Dementsprechend wurden von Microsoft, IBM und SAP drei öffentlich zugängliche UDDI Registries aufgesetzt, die aber seit Januar 2006 alle wieder offline sind bzw. nur noch zu Evaluationszwecken zur Verfügung gestellt werden.

ebXML

Bezüglich ihrer grundlegende Funktionalität ist die *Electronic Business using eXtensible Markup Language (ebXML)* mit UDDI vergleichbar, wobei ebXML allerdings als technisch ausgereifter gilt. Die Entwicklung und Pflege der Spezifikation erfolgte und erfolgt durch Arbeitsgruppen von OASIS und UN/CEFACT; sie wurde darüber hinaus 2004 als ISO 15000 formal standardisiert.

Die ebXML-Spezifikation besteht aus zwei Teilen: Der erste definiert das interne Datenmodell, der andere beschreibt eine SOAP-basierte Web Service Schnittstelle. Eine ebXML-Registry kann zusätzlich zu den Funktionen einer Registry auch als Repository für die beschriebenen Ressourcen fungieren, sie kann aber ohne weiteres auch auf Daten in externen Repositories verweisen. Neben den ggf. vorhandenen Repository-Einträgen speichert eine ebXML-Registry standardisierte Metadaten, um die internen oder externen Repository-Einträge zu beschreiben. Bei diesen Metadaten handelt es sich um die Registry-Einträge im eigentlichen Sinn.

In ebXML sind explizit Föderierungsmechanismen vorgesehen, um Anfragen über mehrere Registry-Instanzen hinweg zu verteilen.

Über den Umfang des tatsächlichen Einsatzes von ebXML liegen keine verlässlichen Zahlen vor; es finden sich nur immer wieder Behauptungen wie die bei Gartner, ebXML „[is] used as much as UDDI“ [AVS05] – also in keinem nennenswerten Umfang.

6.2 Mängel von UDDI und ebXML

Die mangelnde Akzeptanz für UDDI und ebXML lässt sich auf sechs fundamentale Mängel zurückführen:

- Beide Standards legen nicht nur fest, wie Informationen abgefragt werden, sondern sie setzen auf fixe Ontologien mit entsprechenden Datenformaten für die Registry-Einräge. Die Ontologien lassen sich nicht ohne weiteres auf zusätzliche oder grundsätzlich andere Anforderungen hin anpassen, wobei hier ebXML noch etwas flexibler ist als UDDI.
- Beide Standards sind zu umfangreich und zu komplex – ein Ausdruck der Version 3.02 der UDDI-Spezifikation ist mehrere hundert Seiten lang, wozu die XML-Schemata für die APIs und Datenformate noch hinzukommen.
- Beide Standards sind an einen spezifischen Technologie-Stack gebunden, nämlich SOAP-basierte Web Services. Dies betrifft sowohl die definierten Schnittstellen als auch, weniger ausgeprägt, ihr Konzept eines Registry-Eintrages.
- Keiner der Standards verfügt über ein wohldefiniertes Datenaustausch- und Speicherformat für die Registry-Einträge.
- Speziell UDDI setzt implizit ein sehr spezielles organisatorisches Umfeld voraus; Diese Annahmen, die in den definierten Schnittstellen fest verankert sind, passen aber häufig nicht zu den tatsächlichen Anforderungen.
- Wie schon erwähnt bietet insbesondere UDDI keinen hinreichende Unterstützung für die Vernetzung von Registries.

Von diesen in erster Linie technischen Aspekten abgesehen, versäumen beide Standards, das mit der Service-orientierten Architektur, für die sie ja vorgesehen sind, untrennbar verknüpfte Konzept der *Sichtbarkeit* von Diensten mit allen Konsequenzen für die Beschreibung der Dienste, ihren Voraussetzungen und ihrer Semantik aufzugreifen (vgl. dazu auch [ML+06]). Außerdem ignorieren beide Standards weitgehend die Motive, Bereitschaft und Randbedingungen für Firmen und Institutionen, Einträge in eine Registry einzustellen, zu warten und ggf. für die in einer Registry gefundenen Inhalte zu zahlen.

UDDI und, wenn auch weniger ausgeprägt, ebXML vermischen drei wichtige, aber konzeptionell verschiedene Aspekte, die klar unterschieden werden sollten:

- Das Informationsmodell für die Registry-Einträge, letztlich also das Referenzmodell der zu Grunde liegenden Ontologie (vgl. Abschnitt 2.3)
- Die technischen Schnittstellen zur Registry
- Die organisatorischen Strukturen zur Verwaltung und Pflege der Daten in der Registry

Es gibt keinen fundamentalen Grund, weshalb Registries an ein spezifisches Informationsmodell, eine spezifische Technologie wie SOAP-basierte Web Services oder an bestimme organisatorische Strukturen gebunden sein sollte. Jede konkrete Registry muss selbstverständlich auf ein bestimmtes Informationsmodell aufbauen und für seine technischen Schnittstellen eine Auswahl aus den möglichen Alternativen treffen. Aber hier gilt wie so oft: Es ist am besten, die einfachste Lösung zu wählen, mit der die Anforderungen erfüllt werden können.

6.3 Föderierte Registries

Die besten Kenntnisse über konkrete Ressourcen besitzen in aller Regel diejenigen, die die Ressourcen bereitstellen und pflegen – dieses Wissen ist bei verteilten Ressourcen also inhärent dezentral. Langfristig kann also dann mit der besten Qualität der Informationen in einer Registry gerechnet werden, wenn die jeweiligen Registry-Einträge lokal kontrolliert und gepflegt werden. Dies lässt sich am einfachsten realisieren, wenn jeder Resource Provider seine eigene Registry betreibt und die Daten dieser Registries – ad hoc oder regelmäßig – zusammengeführt, die Registries also föderiert werden. Zwar muss man mit einer gewissen Inhomogenität der Beschreibungen rechnen, wenn keine zentrale Instanz strikte Bedingungen zur Wahrung der Gleichförmigkeit der Einträge durchsetzen kann; andererseits spiegelt sich in dieser Inhomogenität auch die Varianz der Phänomene, die von den Registry-Einträgen beschrieben werden sollen.

Dennoch bedarf es eines gewissen gemeinsamen Nenners, eines Mindeststandards für die Beschreibung und Klassifikation von Ressourcen, wenn eine Suche in der Registry-Föderation sinnvolle Ergebnisse liefern soll; andernfalls sind die in den einzelnen Registries beschriebenen Ressourcen wieder nicht sichtbar und die Registry-Föderation erfüllt als Ganzes nicht die Grundanforderung an eine Registry.

Wir sehen deshalb – auch mit Blick auf die Gründe, die wir für das Scheitern von UDDI und ebXML verantwortlich machen – folgende Grundanforderungen an eine Föderation von Registries:

- Trennung der technischen Aspekte des Datenaustauschs von der Organisation der Kollaboration
- Daten sollten in einer sowohl für Maschinen als auch für Menschen lesbaren Form ausgetauscht werden; dadurch wird die Inspektion der Daten, speziell auch zur Qualitätssicherung, ohne aufwändige und für die jeweilige Registry-Implementierung spezifische Zusatzsoftware möglich.
- Die Föderierung darf die Registries nicht an eine bestimmte Technologie wie SOAP-basierte Web Services binden.
- Weil die Registries Ressourcen beschreiben, müssen sie auch in eine Resource Oriented Architecture (ROA) [RR07] passen.
- Die Föderierung muss über verschiedene Kollaborationsmodelle möglich sein, insbesondere sowohl eine hierarchische Föderierung als auch in einem P2P-Modell.

Im *eGovernment Resource Network (eGRN)*[4], das unter Mitarbeit eines der Autoren im Rahmen der CEN/ISSS eGovernment Focus Group [KDM08] entstand, wurde ausgehend von diesen Grundsätzen eine Föderation von Topic Maps-basierten Registries durch Syndikation realisiert, genauer gesagt durch den Austausch von XTM-Fragmenten über einen ATOM-Feed.

7 Zusammenfassung

Für die Speicherung und den Austausch von Ontologien und Instanzdaten kommen letztlich nur RDF+OWL und Topic Maps in Frage; UML ist eher zur Visualisierung geeignet. Die beiden verbleibenden Standards sind hinsichtlich der Beschreibung von Ontologien sowie der zur Verfügung stehenden Abfragesprachen im Wesentlichen gleich mächtig; eine Entscheidung zwischen diesen beiden Standards sollte im Einzelfall davon abhängig gemacht werden, ob die Daten automatisches Schlussfolgern unterstützen sollen – und folglich sehr detailliert und widerspruchsfrei sein müssen – oder ob sie eher auf menschliche Nutzer ausgerichtet sind, die aufgrund ihres Kontextwissens mit Ungenauigkeiten und evtl. Widersprüchen umgehen können.

Speziell mit Blick auf die Einbettung von TextGrid in ein größeres digitales eHumanities-Ökosystem sind sowohl UDDI als auch ebXML nicht als Basis einer Resource Registry geeignet, weil sie weder bzgl. der zu Grunde liegenden Ontologie noch der organisatorischen Strukturen oder der technischen Kollaborationsmodelle hinreichend flexibel sind. Wir haben in Abschnitt 6.3 aufgezeigt, weshalb unserer Ansicht nach eine eHumanities Resource Registry notwendig föderiert sein muss und welche Charakteristika eine – noch zu entwickelnde – Lösung aufweisen muss.

Anhang A: Bibliographie

[AVS05] Abrams, C., Valdes, R., Smith, D. M.: *Core Web Service Standard UDDI Evolves With Version 3.0.2*. Gartner Research ID Number 00126170, 2005. http://www.gartner.com/DisplayDocument?doc_cd=126170

[BB07] Burnard, L., Bauman, S.: P5: *Guidelines for Electronic Text Encoding and Interchange*. Text Encoding Initiative Consortium, 2007. http://www.tei-c.org/Guidelines/P5/

[BÜ08] Büdenbender, S.: *Ontologien und Wortnetze*. TextGrid Report 5.1, 2008. http://www.textgrid.de/fileadmin/TextGrid/reports/TextGrid_Report_5_1.pdf

[CEN06] Blanchon, E., Hopmans, G., Glander, A., Küster, M. W. (eds.): *European Network for Administrative Nomenclature*. CEN Workshop Agreement 15526:2006, ftp://ftp.cenorm.be/PUBLIC/CWAs/e-Europe/ADNOM/CWA15526-00-2006-Mar.pdf

[CHRR04] Clement, L., Hately, A., von Riegen, C., Rogers, T. (eds.): *UDDI Version 3.0.2*. UDDI Spec Technical Committee Draft, OASIS, 2004. http://uddi.org/pubs/uddi_v3.htm

[DNB07] Durusau, P., Newcomb, S., Barta, R. (eds.): Topic Maps Reference Model. ISO/IEC 13250-5 Committee Draft, 2007. http://www.isotopicmaps.org/tmrm/

[GB07] Garshol, L. M., Barta, R. (eds.): Topic Maps Query Language. ISO/IEC 18048 Committee Draft, 2007. http://kill.devc.at/system/files/tmql.pdf

4 http://demo.egrn.eu/

[GR93] Gruber, T. „A Translation Approach to Portable Ontology Specifications.“ In: Knowledge Acquisition 5 (1993), S. 199-220.

[ISO05] ISO/IEC JTC 4 / SC 7: *Information technology -- Open Distributed Processing -- Unified Modeling Language (UML) Version 1.4.2*. ISO/IEC 19501:2005.

[ISO03] ISO/IEC JTC 1 / SC34: *Information technology – SGML applications – Topic maps*. ISO/IEC 13250:2003.

[ISO06] ISO TC 46 / SC 4: *Information and documentation -- A reference ontology for the interchange of cultural heritage information*. ISO 21127:2006.

[ISO06a] ISO/IEC JTC 1 / SC 34: *Information technology – Topic Maps – Part 2: Data model*. ISO/IEC 13250-2:2006.

[ISO07] ISO/IEC JTC 1 / SC 34: *Information technology – Topic Maps – Part 3: XML syntax*. ISO/IEC 13250-3:2007.

[KDM08] Küsters, M. W., Deckers, M., Moore, G.: Final Report of the Project Team of the CEN/ISSS eGovernment Focus Group on the eGovernment Standards Roadmap. CEN/ISSS, 2008. http://www.cen.eu/cenorm/businessdomains/businessdomains/isss/activity/finalreport.pdf

[KLA07] Küster, M. W., Ludwig, C., Aschenbrenner, A.: *TextGrid as a Digital Ecosystem*. In DEST 2007, E. Chang, Ed., 2007.

[KML07] Küster, M. W., Moore, G., Ludwig, C.: *Semantic Registries – Cataloguing eGovernment Resources*. XML-Tage 2007, Berlin.

[MBN05] Moore, M., Bogachev, D., Nishikawa, M. (eds.): Topic Maps Constraint Language. ISO/IEC 19756 Working Draft, 2005. http://www.isotopicmaps.org/tmcl/

[ML+06] MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P., Metz, R.: *Reference Model for Service Oriented Architecture 1.0*. OASIS, 2006. http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf

[OMG07] Object Management Group: *Ontology Definition Metamodel*. OMG 1.0 Beta 1 Specification, 2007. http://www.omg.org/technology/documents/modeling_spec_catalog.htm#ODM

[OS08] OASIS SOA-RM Technical Committee: *Frequently Asked Questions*. http://www.oasis-open.org/committees/soa-rm/faq.php

[PE00] Pepper, S.: *The TAO of Topic Maps – finding the way in the age of infoglut*. Ontopia, 2000. http://www.ontopia.net/topicmaps/materials/tao.html

[PPGU06] Pepper, S., Presutti, V., Garshol, L. M., Vitali, F. (eds.): *Guidelines for RDF/Topic Maps Interoperability*. W3C Editor's Draft, 2006. http://www.w3.org/2001/sw/BestPractices/RDFTM/guidelines-20060630.html

[RR07] Richardson, L., Ruby, S.: *RESTful Web Services*. O'Reilly, 2007.

[W3C04] Kylne, G., Caroll, J. J. (eds.): *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation, 2004. http://www.w3.org/TR/rdf-concepts/

[W3C04a] McGuinness, D. L., van Harmelen, F. (eds.): *OWL Web Ontology Language Overview*. W3C Recommendation, 2004. http://www.w3.org/TR/owl-features/

[W3C08] Prud'hommeaux, E., Seaborne, A. (eds.): *SPARQL Query Language for RDF*. W3C Recommendation, 2008. http://www.w3.org/TR/rdf-sparql-query/