DAML+OIL und OWL.
XML-Sprachen für Ontologien

Referat im Rahmen des Seminars „XML-Technologien“ bei
Prof. Dr.-Ing. Robert Tolksdorf im WS 2002/2003

Jörg Dieckmann*

Februar 2003

*Jörg Dieckmann, Marschnerstr. 40, 12203 Berlin, e-mail: dieck@inf.fu-berlin.de

Inhaltsverzeichnis

1 Motivation für die Idee „Semantic Web“ 3

1.1 Anwendungsgebiete 4

1.2 Warum verwendet man nicht RDF und RDF/S? 5

2 Ontologien 6

2.1 Wozu Ontologien? 6

2.2 Beispiel für eine Ontologie 7

2.3 Beispiel für eine Schlussregel-Anwendung 7

2.4 Zum besseren Verständnis 8

3 XML-Sprachen für Ontologien 8

3.1 historischer Exkurs 8

3.2 DAML+OIL 10

3.3 OWL 14

4 Problemstellungen 15

5 Beispiel-Anwendungen 17

6 Zusammenfassung 19

Literaturverzeichnis 20

Einordnung / Seminarbezug

Das dieser Ausarbeitung zugrundeliegende Referat sollte den mit XML grundlegend vertrauten SeminarteilnehmerInnen eine Einführung in Idee, Geschichte, Technologie und Anwendung des „Semantic Web“ geben. Im Seminar „XML-Technologien“ folgte der dieser Ausarbeitung zugrundeliegende Vortrag einem Referat über RDF und RDF/S. Insofern führe ich hier die entsprechenden Teile nicht erneut aus.

1 Motivation für die Idee „Semantic Web“

Die Idee des „Semantic Web“ (im Folgenden mit SW abgekürzt) beruht grob gesagt auf der Einsicht, dass das herkömmliche WWW mit meist aussschließlich textuell vorliegenden Informationen oftmals unzureichend ist, wenn es um die Klassifizierung der zugrundeliegenden Inhalte geht. Es sind daher schon seit langer Zeit Bemühungen um eine Anreicherung der HTML-Seiten mit diese beschreibenden Meta-Datenfeldern im Gange. Das große Problem, dass sich dabei stellt, ist die computergestüzte Interpretation dieser Meta-Datenfelder, da meist weder Struktur noch Verwendungsbereich und schon gar nicht die semantische Einordnung der zur Beschreibung verwendeten Begriffe festgelegt ist.

Die Idee des SW ist nun, die Struktur der Metadatenfelder nicht festzulegen, sondern sie zu durch die Verwendung benutzerdefinierter Schemata zu flexibilisieren. Diese Anforderung leistet bereits RDF/S, indem Begriffsfelder über semantische Graphen aufgespannt werden, die im Falle RDF bzw. RDF/S über Tripel (Subjekt, Prädikat, Objekt) miteinander in Beziehung gebracht werden, auch die Bildung von Taxonomien[1] wird durch `rdfs:subClassOf` unterstützt.

Durch das Aufspannen von Begriffsfeldern mit entsprechenden Strukturen und das Feststellen der relativen Beziehungen von Begriffen innerhalb dieser wird die maschinelle Bearbeitung von Metadaten und ihrer Zusammenhänge möglich. Die Bereitstellung und Definition solcher Metadaten-Strukturen soll

[1] Bezeichnung für das Herstellen einer systematischen Ordnung durch die Anwendung von festen Regeln.

aber entsprechend der Struktur des WWW ebenfalls vernetzt und möglichst hierarchielos sein. Die damit zusammenhängenden technischen und strukturellen Probleme werde ich im Verlauf der nächsten Abschnitte aufzeigen.

1.1 Anwendungsgebiete

Um die Einführung einer SW-Struktur zu motivieren seien einige Beispiele genannt:

- **Bessere Suchmaschinen / bessere Suchergebnisse**
  Oft ist v.a. bei der Suche nach Wörtern, die verschiedene Bedeutungen gleichzeitig tragen und somit für unterschiedliche Begriffe stehen, mit herkömmlichen Suchmaschinen nicht viel zu erreichen (Bsp.: eine Suche nach „OIL“, einer mit SW zusammenhängenden Sprache führt z.B. zur Homepage der Mineralöl-Firma Mobil). Hier würde eine semantisch genaue Auszeichnung der entsprechenden Seiten mit Meta-Daten und die Möglichkeit der Eingabe eines semantischen Kontextes bei der Suche erheblich schneller zu einem brauchbaren Ergebnis führen.

- **Agenten**
  Nach der Vision von Tim Berners-Lee([Berners-Lee 2001]) sollen Agentensysteme das WWW autonom nach für ihre Aufgaben relevanten Informationen durchsuchen. Dazu ist es notwendig, dass diese die Semantik(en) der von ihnen durchsuchten Informationsquellen kennen - was durch die bereits bekannten (RDF, RDF/S) und neuen, hier vorzustellenden SW-Techniken geleistet werden soll. Im Zuge einer solchen Anreicherung auch dynamisch generierter Seiten um semantisches Markup würde die momentan aktuelle „WebServices“-Entwicklung vermutlich beinahe überflüssig werden, da bei der Informationsbeschaffung aus (X)HTML-Seiten mit SW-Techniken auf die bei WebServices notwendigen Schnittstellenbeschreibungen usw. verzichtet werden könnte (dafür tauchen einige andere Probleme auf).

- **Aufbau und Abbildung von „Community Knowledge“**

Das in vielen Firmen vorhandene „Wissen“ sowohl Einzelner als auch von Arbeitsgruppen oder der gesamten Organisation wird schon seit längerem in verschiedensten Formen abgebildet. Die beim SW eingesetzten Technologien können helfen, dieses Wissen in Form von Ontologien (siehe nächster Abschnitt) in strukturierter Form zugänglich und nutzbar zu machen. Häufig ist entsprechendes Wissen auch bereits in ähnlicher Form vorhanden und kann (wenn damit nicht Geschäftsgeheimnisse preisgegeben werden) relativ direkt zur semantischen Auszeichnung der eigenen internen und externen WWW-Präsenzen verwendet werden (bspw. Produktklassifikationen, Begriffshierarchien, Bauteil-Abhängigkeiten).

1.2 Warum verwendet man nicht RDF und RDF/S?

Es stellt sich die berechtigte Frage, warum die am Aufbau des SW Beteiligten sich gegen die Kombination aus RDF und RDF/S zum Aufbau solcher Strukturen entschieden haben. Offensichtlich ist aber die konzeptionelle Stärke dieser Sprachen für eine Formulierung der notwendigen Strukturen nicht ausreichend:

> „Formal semantics for the primitives defined in RDF Schema are not provided, and the expressivity of these primitives is not enough for full-fledged ontological modeling and reasoning.“ [Broeckstra et al., 2000]

Darum wird mit den im Abschnitt 3 vorgestellten Sprachen eine Schicht definiert, die auf RDF+RDF/S aufbaut und diese um entsprechend mächtigere Konzepte erweitert. Erst diese letzte Schicht wird dann SW genannt:

> „To perform these tasks, an additional layer on top of RDF Schema is needed. Tim Berners-Lee calls this layered architecture the Semantic Web“ [Broeckstra et al., 2000]

2 Ontologien

Als Grundlage zur Abbildung der benötigten Begriffs- und Wissensstrukturen dienen sogenannte *Ontologien*. Der Begriff hat seine Tradition in der Philosophie und ist dort dem erkenntnistheoretischen Bereich zuzuordnen. Der Duden definiert Ontologie als „die Wissenschaft vom Seienden“. Im Zusammenhang mit künstlicher Intelligenz (KI) und der von dieser Forschungsrichtung stark beeinflussten SW-Forschung gibt es nicht mehr *die Ontologie* (bspw. als Wissenschaft) sondern eine Anzahl verschiedener *Ontologien*. Unter einer Ontologie wird in diesem Zusammenhang eine Sammlung und Strukturierung zusammengehöriger Begriffe verstanden. Die in einer Ontologie zusammengeführten Begriffe werden in ihr geordnet, hierarchisiert und miteinander in definierte Beziehungen gebracht. Dementsprechend stellen spezielle Ontologien auch immer eine *Einigung und Festlegung* für Begriffsfelder innerhalb einer Personengruppe dar. Das Gebiet, das mithilfe einer Ontologie erschlossen wird, wird dabei als *Domäne* bezeichnet.

2.1 Wozu Ontologien?

Solche festgelegten, strukturierten Begriffsfelder dienen auch immer dazu, die Kommunikation innerhalb eines Fachgebiets zu erleichtern, indem sie die Verwendungsbereiche von Begriffen festlegen:

> „People can’t share knowledge if they don’t speak a common language“ ([Davenport 1998])

Die Einigung stellt dabei bereits einen wichtigen Aushandlungsprozess dar, dessen Ergebnis (oder Zwischenergebnis) sich in einer Ontologie darstellen lässt. Teilweise werden bereits vorliegende Ontologien auch direkt weiterverwendet, um damit Softwaresysteme zu spezifizieren oder sogar um Code zu generieren.

Eine weitere Möglichkeit, mit Ontologien zu arbeiten, ist das Deduzieren neuer, in den Regeln einer Ontologie bereits „versteckter“ Fakten anhand logischer Schlussregelsysteme (*description logics, frame-based*), die ich hier nicht näher ausführen kann. In der Praxis heisst dies, dass man ähnlich der Benutzung eines PROLOG-Systems Fragen an ein entsprechendes Programm

stellen kann, das anschließend versucht, diese Fragen anhand der in der Ontologie definierten Regeln zu beantworten - oder logische Fehler zu finden (vgl.: OilEd[2] + FaCT[3]).

2.2 Beispiel für eine Ontologie

Eine Beispiel-Ontologie für dieses Referat:

1. Hierarchie:

- Ernährungsweise
  - Fleischfresser
  - Pflanzenfresser
  - Allesfresser

2. Hierarchie:

- Lebewesen
  - Tier
    - Löwe *(Fleischfresser)*
    - Zebra *(Pflanzenfresser)*
  - Mensch
    - Grünkern-Gourmet *(Pflanzenfresser)*
    - Fleisch-Fan *(Fleischfresser)*
    - „Egal-Was-Hauptsache-Lecker“-Typ *(Allesfresser)*

Desweiteren soll gelten, dass Fleisch- und Pflanzenfresser disjunkte Mengen sind, d.h. kein Fleischfresser darf auch Pflanzenfresser sein und umgekehrt. In diesem Beispiel werden also Begriffe sowohl in hierarchische, als auch in nicht-hierarchische Beziehungen gebracht, um einem speziellen Zweck zu dienen (nämlich dem, ein Beispiel zu konstruieren).

2.3 Beispiel für eine Schlussregel-Anwendung

Gegeben seien die folgenden Regeln:

```
(motherOf subProperty parentOf)
(Mary motherOf Bill)
```

[2] `http://oiled.man.ac.uk/`

[3] `http://www.cs.man.ac.uk/horrocks/FaCT/`

wobei `motherOf` und `parentOf` Relationen zwischen Individuen sein sollen und `subProperty` eine Spezialisierung einer Relation angibt. Dann ist durch ein Schlussregelsystem (engl.: *inference engine*) zu ermitteln:

```
(Mary parentOf Bill)
```

2.4 Zum besseren Verständnis

> „An ontology is a formal, explicit specification of a shared conceptualisation.“ [Gruber 1993]

Ontologien enthalten keine Abbilder der Realwelt, sondern (genau wie Softwaresysteme oder die meisten Begriffe wissenschaftlicher Forschung) nur Modelle, in denen sich bestimmte Konzepte ausdrücken (Konzeptualisierung). Ggf. ist die durch eine Ontologie dargelegte *Sichtweise* innerhalb einer Gruppe gemeinsam anerkannt (*shared*).

Modelle sind fast immer vereinfachend, oft auch abstrahierend, um verallgemeinern zu können. Eine Ontologie im Sinne der KI/des SW ist damit eine Formalisierung und Festlegung der Konzepte und Modelle einer bestimmten (Wissens-)Domäne. Durch die Wahl einer geeigneten Sprache zur Modellierung von Ontologien kann mit ihnen maschinell bzw. automatisiert umgegangen werden (siehe nächster Abschnitt).

3 XML-Sprachen für Ontologien

3.1 historischer Exkurs

Die erste (für das WWW bzw. XML relevanten) Sprache zur Formulierung von Ontologien war `SHOE`[4] (Simple HTML Ontology Extension). Mit `SHOE` versuchten seine Erfinder, die Technik des semantischen Markups mithilfe von Ontologien so einfach wie möglich zu gestalten, indem sie die Anmerkungen und Ontologien direkt als HTML-Erweiterungen definierten (vgl. [Luke et al. 1996]).

Komplexere, konzeptionell stärkere Ansätze wurden mit `DAML` und `OIL` verfolgt:

[4] http://www.cs.umd.edu/projects/plus/SHOE/

DAML Die *DARPA Agent Markup Language* wurde im Rahmen eines Forschungsprogramms der *Defense Advanced Research Projects Agency* entwickelt. Die dem US-Verteidigungsministerium unterstellte DARPA befasst sich nach Redbrake (Redbrake 2001[5]) mit Forschungsprojekten zum Erreichen eines technischen Fortschritts der „weit vor der technischen Evolution liegt".

Das DAML-Programm wurde im August 2000 gestartet, seine Ergebnisse sind aufgrund des „unclassified"-Status' des Projekts allgemein zugänglich, werden aber von der DARPA durchaus auf den militärischen Anwendungszweck hin verwendet.

OIL Bereits früher, im August 1999 begonnen, wurden ähnliche Bemühungen, die aber von ihrer Zielsetzung her eher dem E-Commerce-Hype der 1990er entsprangen, im Programm „On-To-Knowledge"[6] durch die EU unterstützt[7]. Die Sprache OIL war zwar für die Formulierung von Ontologien geeignet, jedoch nicht XML-konform und für die Integration in (X)HTML-Seiten ungeeignet.

DAML+OIL Sowohl DAML als auch OIL waren von Beginn an auf einem hohen Abstraktionsniveau angesiedelt, jedoch war OIL im Gegensatz zu DAML von Beginn an auf die Formulierung von Ontologien ausgerichtet, DAML erst in der speziellen Version DAML-ONT. Offensichtlich erkannten die Beteiligten jedoch bald die Gemeinsamkeiten der beiden Projekte und spezifizierten als Zusammenfassung beider Sprachen `DAML+OIL` (letzter Language Release im März 2001[8]). Zum momentanen Zeitpunkt (Februar 2003)

[5] `wwwmath.uni-muenster.de/u/lammers/EDU/ws01/ Softwareagenten/Referate/A32.Redbrake.pdf`

[6] `http://www.ontoknowledge.org`

[7] The overall objective of IAF as set out in the IST Specific Programme is to develop: „...advanced technologies for the management of information content to empower the user to select, receive and manipulate (in a manner that respects the user's right to privacy) only the information required when faced with an ever increasing range of heterogeneous sources." This includes: „... improvements in the key functionalities of large-scale multimedia asset management systems (including the evolution of the World Wide Web) to support the cost effective delivery of information services and their usage." (von `http://www.cordis.lu/ist/ka3/iaf/index.htm`)

[8] `http://www.daml.org/2001/03/daml+oil-index.html`

scheint DAML+OIL als Standard für Ontologien und semantisches Markup zu gelten.

OWL Die „W3C Web Ontology Working Group“ führte (als Abschluss des historsichen Exkurses) die „Web Ontology Language“ (OWL) auf Basis von DAML+OIL ein. (letzter OWL Working Draft Nr.8 von 11/2002). Es haben sich nur kleine Änderungen ergeben – der Grund, warum das W3C die Sprache umbenennen und respezifizieren musste (und warum sie nicht WOL heisst), ist mir unklar.

3.2 DAML+OIL

Einführung DAML+OIL ist keine komplette Neuentwicklung, sondern baut (wie DAML) auf RDF und RDF/S auf. Damit sollte erreicht werden, dass RDF-basierende Anwendungen mit DAML+OIL-ausgezeichneten Seiten so wenig Probleme wie möglich bekämen. Die Syntax von DAML+OIL ist daher selbst in RDF/S formuliert[9].

Elemente der DAML+OIL-Syntax (eine offizielle Einführung findet sich im „DAML+OIL walktrough“[10])

- Eine Ontologie anlegen:

```
<daml:Ontology rdf:about="Name der Ontologie">
    <daml:versionInfo>
        $Id: daml+oil-ex.daml,v 1.4 2001/01/... (CVS-Stil)
    </daml:versionInfo>
    [   ggf. DC-Tags ]
    <daml:imports rdf:resource= "http://..."/>
</daml:Ontology>
```

  Mit dem obigen Quellcode wird eine Ontologie in DAML+OIL angelegt, deren Name mit `rdf:about=` angegeben werden kann. Über den

[9] vgl. `http://www.daml.org/2001/03/daml+oil`

[10] `http://www.daml.org/2001/03/daml+oil-walkthru`

<versioninfo>-Tag kann eine Versionsinformation im CVS-Stil hinterlassen werden. Der hier mit [DC] gekennzeichnete Teil kann weitere Meta-Informations-Tags im Dublin-Core-Standard enthalten (bspw. <dc:author>). Mit dem <daml:imports>-Statement kann auf bestehende Ontologien Bezug genommen werden, deren URI per rdf:resource= angegeben wird.

- Klassen

```
<daml:Class rdf:ID="">
    <daml:about=""/>
    <rdfs:subClassOf/>
    <daml:disjointWith/>
    <daml:disjointUnionOf/>
    <daml:sameClassAs/>
</daml:Class>
```

Mit dem <daml:Class>-Tag wird eine Klasse angelegt (ursprünglich wurde hier <rdfs:Class> verwendet, teilweise tun Editoren/Generatoren das auch noch), deren Name entweder durch <daml:Class rdf:ID= "Name"> oder <daml:about="name"/> angegeben wird.

Das <rdfs:subclassOf>-Konstrukt kann null , eine oder mehrere Klassen enthalten, von denen die aktuelle Klasse Unterklasse ist (nota bene: hier wird noch rdfs verwendet!). <daml:disjointWith> dient zur Angabe von disjunkt-Beziehungen im Sinne der Mengenlehre zwischen Klassen und <daml:disjointUnionOf> zur Angabe von paarweise disjunkten Klassen. Der <daml:sameClassAs>-Tag gibt eine Entsprechung von Klassen an und kann somit zum Herstellen von Beziehungen zwischen verschiedenen Ontologien verwendet werden.

- Klassenkonstrukte
  Neue Klassen können auch durch Verknüpfungen bestehender erstellt werden:
  - <daml:intersectionOf>...</daml:intersectionOf> definiert einen Schnitt zwischen zwei Klassen,

– `<daml:unionOf>` eine Vereinigung zweier Klassen

– und `<daml:complementOf>` das Komplement einer Klasse.

Außerdem können Klassen durch Aufzählung ihrer Elemente erstellt werden, die allerdings Instanzen sein müssen (dazu später mehr). Beispiel:

```
<daml:oneOf parseType="daml:Collection">
    <daml:Thing rdf:about="#Verweis1">
    <daml:Thing rdf:about="#Verweis2">
    ...
    <daml:Thing rdf:about="#VerweisN">
</daml:oneOf>
```

- Properties
  Mit `<daml:ObjectProperty>` kann eine Relation zwischen Klassen angegeben werden. Dabei muss innerhalb des Tags mit `<rdfs:domain rdf:resource="...>` per URI angegeben werden, welche *Urbildmenge* (in Form von Klassen) und mit `<rdfs:range ...>` welche *Bildmenge* jeweils zulässig ist. Beispiel:

```
<daml:ObjectProperty rdf:ID="hasFather">
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="#Male"/>
</daml:ObjectProperty>
```

  definiert eine Beziehung „hatVater“ zwischen einer Person und einem Mann.

  `<daml:DatatypeProperty>` fügt einer Klasse „Attribute“ bestimmter XMLSchema-Datentypen hinzu, die per Referenz der entsprechenden Schema-Definition innerhalb des `<rdf:range ..>`-Tags angegeben werden.

  Für beide Arten von Properties kann außerdem der Typ des Attributs angegeben werden: dazu stehen `<daml:UniqueProperty>` (muss innerhalb der Klasse eindeutig sein) `<daml:UnambiguousProperty>` (darf

nur zu einer Klasse gehören) und `<daml:TransitiveProperty>` (selbsterklärend) zur Verfügung[11].

- Restrictions
  Einschränkungen können Klassen und Relationen so beschränken, dass Spezialfälle behandelt werden können. Dazu wird der `<daml:Restriction>`-Tag verwendet. Beispiel:

```
<rdfs:subClassOf>
    <daml:Restriction daml:cardinality="1">
        <daml:onProperty rdf:resource="#hasFather"/>
    </daml:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
    <daml:Restriction>
        <daml:onProperty rdf:resource="#shoesize"/>
        <daml:minCardinality>1</daml:minCardinality>
    </daml:Restriction>
</rdfs:subClassOf>
```

  Hier werden (für eine Personen-Klasse) zwei anonyme Klassen definiert, von denen diese Klasse erben soll. Sie schränken die Relationen `hasFather` und `shoesize` (die ggf. in einer höheren Hierarchieebene außerhalb der Personen-Klasse definiert sind) so ein, dass jede Person genau einen Vater und mindestens eine Schuhgröße hat.

  Dabei gibt `<daml:onProperty>` an, um welche Relation bzw. welches Attribut es sich handelt, weitere Einschränkungen können über `<daml:hasClass>` (fordert eine bestimmte Klasse als Ziel der Relation) und `<daml:toClass>` (schränkt die Restriktion auf die lokale Klasse ein). Zusätzlich zu den beiden im Beispiel gezeigten Kardinalitäts-Beschränkungen (`cardinality, minCardinality`) stehen noch weitere (`maxCardinality`) sowie *qualifizierte* Kardinalitäten zur Verfügung (in Form bestimmter XSD-Datentypen).

- Instanzen
  Damit die Klassen auch semantisch sinnvoll eingesetzt werden können, erzeugt man Instanzen der definierten Klassen:

[11] Symmetrische Relationen führt erst OWL ein! Vgl. Abschnitt 3.3

```
<Person rdf:ID="Peter">
  <shoesize>9.5</shoesize>
  <age>46</age>
  <shirtsize>15</shirtsize>
</Person>
```

(hier als Beispiel einer Person). Alternativ kann man die Instanzen auch mit `<rdf:Description rdf:about="...` anlegen.

3.3 OWL

OWL (Web Ontology Language), die von der Web Ontology working group des W3C herausgegebene Sprache, ähnelt DAML+OIL sehr stark. Sie liegt in drei Versionen vor (OWL Lite, OWL DL, OWL Full), die sich in der Anzahl der Syntaxelemente unterscheiden (OWL Lite vs. OWL DL/Full). Das W3C gibt hier als Begründung für die Trennung an, OWL Lite vorgesehen zu haben, um einfachere Implementierungen zu unterstützen.

So werden einige Tags umbenannt, einige Bedeutungen verändert und nur wenige hinzugenommen (neu sind hier v.a. symmetrische Relationen). Es folgt eine kurze Auflistung:

```
DC-Tags, versionInfo, imports
Class, rdfs:subClassOf, sameClassAs
rdf:Property, rdfs:subPropertyOf, samePropertyAs
rdfs:domain, rdfs:range
Individual, sameIndividualAs differentIndividualFrom
inverseOf
TransitiveProperty, SymmetricProperty
FunctionalProperty (unique)
InverseFunctionalProperty (unambiguous)
allValuesFrom (previously toClass)
someValuesFrom (previously hasClass)
minCardinality (restricted to 0 or 1)
maxCardinality (restricted to 0 or 1)
cardinality (restricted to 0 or 1)
```

OWL DL/Full fügen außerdem noch folgende Tags hinzu:

```
oneOf (enumerated classes)
disjointWith
unionOf, intersectionOf, complementOf
minCardinality, maxCardinality, cardinality
```

4 Problemstellungen

Da die SW-Forschung noch relativ jung ist (DAML bspw. wurde erstmals 2000 entwickelt), sind auch noch viele Probleme ungelöst. Einige der offenen Fragen möchte ich hier kurz vorstellen:

- **keine global verbindlichen Ontologien**
  Durch die verteilte, nicht-hierarchische Struktur des WWW können viele gleichartige oder ähnliche Ontologien zu einer Domäne entstehen. Ein „vernünftiges“ SW muss mit dieser Diversifizierung umzugehen lernen: es müssen Beziehungen zwischen den Klassen verschiedener Ontologien aufgebaut werden (einfachster Fall: sameClassAs) - v.a. auch zwischen denen verschiedener Sprachen (Ein Ansatz dazu ist das Project OntoMerge vgl. [McDermott/Dou/Qi 2003], das die Erledigung dieser Aufgabe so weit wie möglich automatisieren soll). Die Probleme dabei sind vielfältig, v.a., wenn es um ähnliche aber nicht gleiche Konzepte geht. Dabei stellt sich bspw. die Frage, wie man Differenzen zwischen Konzepten explizit in einer entsprechenden Beziehung formulieren kann.

- **Qualität von benutzerdefinierten Ontologien**
  Zur Erstellung von qualitativ hochwertigen, widerspruchsfreien Ontologien ist Know-How bzw. technische Unterstützung notwendig (z.B. OntoEdit mit FaCT[OilEd/FaCT]), die Frage ist, wie gut eigene Ontologien werden und ob, wie und wie weit sie kommuniziert werden. Daher ist ein verbreiteter Ansatz bei der Erstellung von Ontologien die Wiederverwendung und Anpassung vorhandener. Das Problem der *Vertrauenswüdigkeit* und *Einschätzung der konzeptionellen Vollständigkeit* bzw. der konzeptionellen Stärke bleibt bisher auch ungelöst.

- **Agenten: Ich weiss, dass ich nicht alles weiss**
  Agenten und Schlussfolgerungssysteme, die mit Ontologien arbeiten, dürfen weder Annahmen machen, die über das beschränkte Wissen der ihnen zur Verfügung stehenden Ontologien hinausgehen, noch dürfen sie annehmen, dass sie damit in jedem Fall über eine Domäne vollständig informiert wären, da ggf. andere Konzepte existieren, über die sie keine

Kenntnis haben, oder zu deren Ontologien ihnen keine „Übersetzungen“ (bridging axioms/mappings, vgl. [McDermott/Dou/Qi 2003]) zur Verfügung stehen.

- **Wie genau werden die Begriffe im Markup eingesetzt?**
  Auch bei klar definierten Begrifflichkeiten innerhalb einer Ontologie ist nicht sicher, ob die Autoren von Seiten für das SW diese auch an den richtigen Stellen und im richtigen Kontext einsetzen - hier liegt eine große Quelle von Unsicherheiten. Anleitungen zum Einsatz der Begriffe innerhalb der Ontologien wären notwendig.

- **Wie geht man mit komplexen/unscharfen Begriffen um?**
  In nicht Natur- oder Strukturwissenschaftlichen Bereichen kommt es häufig vor, dass sich nicht einmal die Wissenschaftler eines Landes oder gar einer Fachrichtung auf eine präzise gemeinsame Definition eines Begriffs einigen können. Nur die Randbedingungen sind evtl. die gleichen (z.B.: „Bildung“ „Lernen“, „Wissen“ innerhalb der Erziehungswissenschaften) Inwieweit können durch DAML+OIL/OWL-Ontologien auch unscharfe Begriffe so formuliert werden, dass man sie für semantisches Markup ausreichend definiert hat?

- **Entwicklung von neuen Ontologien**
  Eine besonders große Bedeutung kommt der Entwicklung von neuen Ontologien zu. Inwieweit erfordert diese Tätigkeit weitergehende Qualifikationen bspw. aus dem KI-Bereich? Kann diese Tätigkeit durch Tools so unterstützt werden, wie es heute bspw. im Web-Authoring-Bereich der Fall ist? Oder sollten „Laien“ generell nur auf Ontologien, die von „Experten“ erstellt wurden, aufbauen? Ein konkreter Ansatz dazu findet sich bei [Maedche/Staab 2001]: mithilfe von NLP[12]-Technologie versuchen die Autoren, Konzepte und Strukturen für die Konstruktion neuer Ontologien aus vorhandenem Material abzuleiten (mithilfe der *TextToOnto*[13]-Software).

[12] Natural Language Processing
[13] http://ontoserver.aifb.uni-karlsruhe.de/texttoonto/

- **Versionsmanagement**
  Innerhalb des nicht-hierarchischen WWW muss der Veränderung von Begriffen oder ganzer Domänen Rechnung getragen werden – es wird ein Versionsmanagment für Ontologien benötigt, damit Seiten, die sich auf vorherige Versionen eine bestimmten Ontologie beziehen, dadurch nicht plötzlich unbrauchbar werden (vgl. Ansätze des W3C - Jahres und Monatszahlen in URLs). Hierzu haben die DAML+OIL-Entwickler bereits den `<daml:version>`-Tag vorgesehen, wie genau eine Versionierungsstrategie für Ontologien aussehen soll (v.a., wenn sie verteilt entwickelt, ergänzt und bearbeitet werden), ist noch unklar.

5 Beispiel-Anwendungen

Folgende Beispielanwendungen habe ich innerhalb des dieser Ausarbeitung zugrundeliegenden Vortrags benutzt:

- Vorführung eines Ontologie-Editors am Beispiel OILEd (s. [OilEd/FaCT]) mit der Beispiel-Ontologie über Menschen, Tiere, Fleisch- und Pflanzenfresser (s.S. 7, Abschnitt 2.2),

- Vorführung eines Ontologie-gestützten Mozilla-PlugIns (COHSE/DLS[14], das nach Schlüsselwörtern in betrachteten HTML-Seiten sucht und Tooltip-artige Anmerkungen aus der verwendeten Ontologie hinzufügt,

- Vorführung der *OntoMat-Annotizers*[15]-Software, die dazu dient, möglichst einfach Ontologie-gestütztes semantisches Markup auf DAML+OIL-Basis in HTML-Seiten zu integrieren,

- Vorführung von *TAP*[16], einer „activity-based search-engine“, der eine Ontologie über Künstler, Musiker und Schauspieler zugrunde liegt,

- Erwähnung des *Gene Ontology Consortiums*[17], das eine fortwährend

[14] http://cohse.semanticweb.org/software.html
[15] http://annotation.semanticweb.org/ontomat/index.html
[16] http://tap.stanford.edu:8000/tap
[17] http://www.geneontology.org/

ergänzte und bearbeitete Ontologie über die Bedeutung genetischer Codes betreut (bisher nicht DAML+OIL-basiert),

- Erwähnung des *Open Directory Projects*[18], in dem versucht wird, ein verteilt betreutes, nicht-kommerzielles Verzeichnis des WWW aufzubauen,

- Erwähnung des *Standard for Upper Ontologies*[19]-Projekts der IEEE, das sich mit der Erstellung einer Ontologie für möglichst allgemeingültige Meta-Konzepte beschäftigt, auf deren Basis andere, spezielle Ontologien aufgebaut werden sollen

- und die Erwähnung einer Anwendung namens *DAML-Map*[20], eines Ontologie-basierten Landkartengenerators.

[18] http://dmoz.org

[19] Zitat: „An upper ontology is limited to concepts that are meta, generic, abstract and philosophical, and therefore are general enough to address (at a high level) a broad range of domain areas. Concepts specific to given domains will not be included; however, this standard will provide a structure and a set of general concepts upon which domain ontologies (e.g. medical, financial, engineering, etc.) could be constructed.“ URL: http://suo.ieee.org/

[20] http://www.daml.org/2001/06/map/

6 Zusammenfassung

1. Das Semantic Web soll das klassische WWW durch ein mit Metadaten hoher Qualität angereichertes Web ersetzen.

2. *Ontologien* bilden die Basis für „benutzerdefinierte“ Metadaten-Konzepte

3. DAML+OIL/OWL scheinen sich als Standard für die Formulierung von Ontologien für das SW herauszubilden. Es existieren bereits viele nutzbare Ontologien.

4. Die Verbreitung von semantischem Markup ist bisher (2/2003) noch sehr gering. Viele mit dem SW entstehenden Probleme sind noch zu lösen, es existieren aber bereits vielversprechende Fallstudien.

5. SW-Problemstellungen und -technologien sind momentan ein großer, schnell wachsender Forschungsbereich von zukünftig höchstwahrscheinlich hoher Bedeutung.

Literatur

[Hinweis:] Alle Links wurden von mir im Februar 2003 geprüft und funktionierten zu diesem Zeitpunkt

[Berners-Lee 2001] Berners-Lee, Tim; Hendler, James; Lassila, Ora: The Semantic Web In: Scientific American, 284, 5, pp. 34-43, 2001.

[Berners-Lee 1998] Berners-Lee, 1998. Berners-Lee, T. (1998). Semantic web road map. Internal note, World Wide Web Consortium. http://www.w3.org/DesignIssues/Semantic.html.

[Broeckstra et al., 2000] J. Broekstra/M. Klein/S. Decker/D. Fensel/I. Horrocks: Adding formal semantics to the web: building on top of rdf schema. In: Proceedings SemWeb 2000.
http://citeseer.nj.nec.com/broekstra00adding.html

[Davenport 1998] T. Davenport, L. Prusak: Working Knowledge. Harvard Business School Press, 1998

[Gruber 1993] T. R. Gruber: Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In: Formal Ontology in Conceptual Analysis and Knowledge Representation, Kluwer Academic Publishers, Deventer, The Netherlands. Hrsg.: N. Guarino/R. Poli,1993
http://citeseer.nj.nec.com/gruber93toward.html

[Luke et al. 1996] Luke, S., Spector, L., Rager, D. (1996). Ontology-based knowledge discovery on the worldwide web. In Proceedings of the Workshop on Internet-based Information Systems, AAAI'96. 1996
http://citeseer.nj.nec.com/article/luke96ontologybased.html

[McDermott/Dou/Qi 2003] Ontology Translation and Translating Ontologies on the Semantic Web
http://cs-www.cs.yale.edu/homes/dvm/papers/wwwont.pdf
WWW2003, May 20-24, 2003, Budapest, Hungary.
http://cs-www.cs.yale.edu/homes/dvm/daml/
Ontology translation by ontology merging and automated reasoning. In *Proc. EKAW Workshop on Ontologies for Multi-Agent Systems.*
http://cs-www.cs.yale.edu/homes/dvm/papers/DouMcDermottQi02.ps

[OilEd/FaCT] OilEd, ein Editor für Ontologien in OIL, DAML+OIL u.v.a.
http://oiled.man.ac.uk/

[Maedche/Staab 2001] Maedche, Alexander/Staab, Steffen: Learning Ontologies for the Semantic Web
Semantic Web Workshop 2001 Hongkong, China
http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-40/maedche+staab.pdf

[DAML-Projekt] http://www.daml.org
DAML+OIL-Standard unter:
http://www.daml.org/2001/03/daml+oil-index

[Semantic Web allgemein] http://www.semanticweb.org

[OIL / OnToKnowledge] http://www.ontoknowledge.org

[W3C WebOntology WG / OWL] http://www.w3.org/2001/sw/WebOnt/