

Methoden der Wirtschaftsinformatik/Systemanalyse Ausarbeitung

Semantic Web

XML und RDF



Inhalt

1. Einleitung.....	3
2. XML – Die eXtensible Markup Language	4
2.1 Kurze Auflistung wichtiger Merkmale der XML.....	4
2.2 XML als Markup- und Meta-Sprache.....	5
2.3 Aufbau eines XML-Dokuments	6
2.3 DTDs – Document Type Definitions und XSD – XML-Schema-Definition	8
2.4 Ist XML alleine ausreichend für das Semantic Web?	10
3. Ontologien in RDF	11
3.1 RDF – Einführung	11
3.2 URI – Uniform Resource Locator	12
3.3 Literale	12
3.4 Vom Graph zum Tripel	13
3.5 Serialisierung in XML	15
3.6 Der W3C RDF Validator.....	17
3.7 Vokabular	17
4. Zusammenfassung und abschließende Bemerkung.....	18
5. Abbildungsverzeichnis	19
6. Quellenverzeichnis	20
7. Anhang	22
7.1 Screenshot des W3C Validators	22

1. Einleitung

Das Internet hat sich in den letzten Jahren zu einem immer wichtiger werdenden Medium entwickelt. Firmen nutzen Ihre Webpräsenz um sich vorzustellen und bewerben bei dieser Gelegenheit ihre eigenen Produkte. Doch nicht nur Firmen sind im Internet aktiv, viele Privatpersonen besitzen ebenfalls eine eigene Internetpräsenz und schreiben in Blogs über ihren Alltag, über Erfahrungen mit Dienstleistungen und Produkten und über alles andere, was sie für lesenswert halten. Die Technik, die in diesen Blogs zum Einsatz kommt, gehört zum Themengebiet des sogenannten **Web 2.0**. Dabei handelt es sich um einen künstlich erschaffenen Begriff, der den aktuellen Wandel des Internets beschreiben soll. Früher wurden Inhalte fast ausschließlich durch statische Seiten vermittelt; der Gedanke der Eigeninitiative tritt nun aber in den Vordergrund. Das wohl bekannteste Beispiel hierfür ist Wikipedia. Dort werden die Inhalte ausschließlich durch Benutzer bereitgestellt und die Wikimedia Foundation stellt lediglich die Plattform und Technik zur Verfügung. Wer sich bereits mit dem Thema auseinandergesetzt hat weiß, dass die Organisation von Informationen in solch einem Umfang schnell unübersichtlich werden kann und diese teilweise redundant vorkommen können. Eine gezielte Suche nach Wissen wird somit durch die Masse an Informationen erschwert. Diesem Problem kann man mit semantischen Verknüpfungen entgegenwirken. Im Folgenden soll nun näher erläutert werden, wie diese Verknüpfungen in Form von Meta-Informationen für einen Computer leserlich dargestellt werden können.

2. XML – Die eXtensible Markup Language

In diesem Kapitel steht die Sprache XML im Vordergrund. Sie stellt eine syntaktische Grundlage für das Semantic Web dar. Die Syntax der Sprache RDF, welche für das Semantic Web von großer Bedeutung ist, beruht auf XML.

2.1 Kurze Auflistung wichtiger Merkmale der XML

Im Folgenden sind einige wichtige Merkmale von XML aufgelistet, welche zu einem späteren Zeitpunkt näher erläutert werden¹.

XML steht für eXtensible **M**arkup **L**anguage. Eine mögliche Übersetzung wäre „erweiterbare Auszeichnungssprache“.

XML kann andere Auszeichnungssprachen erfinden, zum Beispiel XHTML. Hierbei wird HTML durch eine XML Syntax beschrieben. Deshalb bezeichnet man XML als **Meta-Sprache**.

Eine Dokumentationstyp-Definition (DTD: **D**ocument **T**ype **D**efinition) begrenzt den nutzbaren Namensraum von Tags. Sobald sich ein Dokument an die vorher definierte DTD hält, nennt man es **valide**.

XML unterscheidet zwischen Groß- und Kleinschreibung und ist somit **case-sensitive**.

Ein **wohlgeformtes** XML-Dokument hält sich an alle XML-Regeln. Ein Parser kann die Korrektheit der Grammatik überprüfen. Jedes XML-Dokument **muss** wohlgeformt sein.

Beispiele wichtiger XML-Regeln:

- Das Dokument besitzt genau ein Wurzelement.
- Alle Elemente besitzen ein Anfangs- und eine Endkennung. Elemente ohne Inhalt können in sich geschlossen sein.
- Ein Element darf nicht mehrere Attribute mit dem gleichen Namen besitzen.

XML verwendet **Unicode** Zeichenkodierung.

Um ein XML-Dokument zu nutzen, muss eine Anwendungsschnittstelle vorhanden sein. Das Dokument an sich ist nicht ausführbar, sondern enthält lediglich

Informationen, die von anderen Anwendungen genutzt werden können. Ein spezieller Parser muss hierbei genutzt werden.

2.2 XML als Markup- und Meta-Sprache

XML ist eine sogenannte „**Recommendation**“ (zu Deutsch: Empfehlung) des W3C (**World Wide Web Consortium**) für eine bestimmte Art maschinenlesbarer Dokumente. Die Recommendation des W3C beschreibt XML Dokumente und teilweise das Verhalten der Software, welche diese Dokumente weiterverarbeitet.

In der Informatik werden Auszeichnungssprachen genutzt, um Teile eines Dokuments mit zusätzlichen Informationen zu versehen. Das Auszeichnen von Informationen wird als **Annotation** bezeichnet, die Informationen an sich werden **Metadaten** genannt. Metadaten dienen zur Beschreibung von anderen Daten. Auch HTML (**Hypertext Markup Language**) ist eine **Markup** Sprache, welche zusätzliche Informationen in Tags zur Darstellung im Dokument hinterlegt. Beim Besuchen einer Webseite wertet der Browser die Tags aus und formatiert entsprechend der Annotation den Text bzw. stellt verschiedene Gruppierungselemente wie Tabellen oder Listen dar. HTML Tags können beispielsweise folgendermaßen aussehen:

In HTML Dokumenten kann man Text `<i>kursiv</i>` oder auch `fett` darstellen.

Dieser Text wird in standardkonformen Webbrowsern wie folgt dargestellt:

In HTML Dokumenten kann man Text *kursiv* oder auch **fett** darstellen.

Anders als HTML verwendet XML Tags nicht zur Auszeichnung der Darstellung von Dokumenten, sondern zur Festlegung einer **logischen Struktur**.

XML ist eine Meta-Sprache zur Erstellung von Markup-Sprachen. XML definiert nicht eine Menge von bestimmten Tags, sondern erlaubt das allgemeine Definieren von Markup-Sprachen. Während mit HTML Texte formatiert und dargestellt werden können, werden mit XML Sprachen für Dokumente mit komplett unterschiedlichen Zwecken definiert.

Ein Beispiel für eine XML Annotation:

```
Das Buch <buch>Harry Potter</buch> wurde von <autor>Joanne K.  
Rowling</autor> geschrieben.
```

XML bietet eine vielseitige Möglichkeit Daten aller Art für den Computer lesbar zu speichern. Selbst komplexe Datenmodelle können in ein XML-Dokument überführt werden. Man spricht hierbei von **Serialisierung** (bzw. die Daten werden **serialisiert**). Im Microsoft .NET Framework können sogar ganze Objekte mit ihren Zuständen in XML-Dokumente abgelegt und bei Bedarf erneut ausgelesen und ausgewertet werden.

Die grundlegende Struktur eines jeden XML-Dokuments sowie die wichtigsten Sprachmittel werden nun im Folgenden näher dargestellt.

2.3 Aufbau eines XML-Dokuments

XML-Dokumente beginnen mit einer einleitenden **Deklaration**, die das Dokument als XML-Dokument bezeichnet und sowohl die Versionsnummer, als auch den verwendeten XML-Standard enthält. Optional befindet sich dort auch eine Festlegung der Zeichenkodierung. Eine Deklaration sieht beispielsweise wie folgt aus:

```
<?xml version="1.0" encoding="utf-8"?>
```

XML-Dokumente enthalten verschiedene **Ausdrücke**. Das **Element** wird am meisten genutzt. Sie beginnen grundsätzlich mit einem Start-Tag und enden mit einem Ende-Tag. Jedes Tag besteht aus einem XML-Namen, welcher von spitzen Klammern umschlossen wird. Die Daten und Elemente zwischen dem Start-Tag und dem Ende-Tag sind der eigentliche **Inhalt**. Ein kleines Beispiel:

```
<student>Hans Müller</student>
```

Wie im Beispiel zu sehen ist, schließt das Ende-Tag das Start-Tag. Alternativ zu dieser Notation kann auch ein **selbstschießendes** Tag verwendet werden. Anstatt des Inhalts können weitere, beliebig oft verschachtelte Elemente stehen.

Jedes XML-Dokument darf nur ein **Wurzelement** besitzen, welches direkt nach der Deklaration stehen muss. Alle weiteren Elemente müssen innerhalb dieses

Wurzelementes stehen. Dies ist eine Grundbedingung, dass das Dokument wohlgeformt sein kann.

Für XML-Namen gilt die Regel, dass diese entweder mit einem Unterstrich oder mit einem Buchstaben anfangen müssen. Zahlen dürfen im Namen enthalten sein, aber nicht an erster Stelle stehen. Die Länge kann beliebig gewählt werden, wobei aber zwischen Groß- und Kleinschreibung unterschieden werden muss. Somit ist Student ungleich student und ungleich sTuDeNt usw.

Start-Tags und selbstschließende Tags können Attribute besitzen. Hierbei kann einem bestimmten XML-Element ein Wert zugewiesen werden. Jedes Attribut besitzt einen Namen, der den gleichen Beschränkungen unterliegt wie der Name eines Tags. Die Anwendung eines Attributs in einem XML-Dokument lässt sich am besten durch ein Beispiel darstellen:

```
<!-- Tag mit einem verschachtelten Tag, welches Daten als Inhalt enthält -->
<Mensch>
  <Student>Hans Mustermann</Student>
</Mensch>

<!-- Tag mit einem Attribut und Daten als Inhalt -->
<Mensch Position="Student">Hans Mustermann</Mensch>

<!-- Selbstschließendes Tag mit zwei Attributen -->
<Mensch Position="Student" Name="Hans Mustermann" />
```

An dieser Stelle sei erwähnt, dass ein Text, der durch `<!--` und `-->` eingeschlossen ist, als Kommentar behandelt wird.

In XML gibt es ein spezielles Attribut, um die Sprache des Dateninhalts eines Elements zu spezifizieren. Dieses Attribut heißt **xml:lang** und könnte in unserem Beispiel den Wert **de** besitzen. Die üblichen Ländercodes sind hier die möglichen Werte.

Abschließend lässt sich sagen, dass XML eine **Baumstruktur** als Datenmodell nutzt. Tags entsprechen den Knoten, jedes weitere verschachtelte Tag entspricht einem Unterknoten. Um die Struktur eines XML-Dokuments weiter vorzugeben, kommen DTDs oder XML-Schema-Dokumente zum Einsatz. Diese werden im Kapitel 2.3 näher erläutert.

2.3 DTDs – Document Type Definitions und XSD – XML-Schema-Definition

DTDs und XML-Schema fordern zusätzlich zu den Regeln der „Wohlgeformtheit“ eines XML-Dokuments weitere Einschränkungen. Oft ergeben sich diese Regeln durch semantische Gegebenheiten der Problemstellung. In einer DTD oder in einem XML-Schema kann die für eine Anwendung oder einen Bereich vorgesehene Struktur von XML-Dokumenten vorgegeben werden. Es wird eingeschränkt, welche Element- und Attribut-Namen zulässig sind, wie Tags verschachtelt und Elemente und Attribute verwendet werden dürfen. Mit Hilfe eines Parsers, es handelt sich hierbei um sogenannte **validierende** XML-Parser, kann zusätzlich zur Wohlgeformtheit die Gültigkeit in Bezug zu einer vorgegebenen Struktur überprüft werden. Ein Beispiel einer DTD, nämlich für XHTML, ist im Quellenverzeichnis unter der Nummer 6 zu finden.

Ein DTD für das Beispiel im vorhergehenden Kapitel könnte demnach so aussehen:

```
<!ELEMENT Menschheit (Mensch+)>
<!ELEMENT Mensch (Name?, Position?)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Position (#PCDATA)>
<!ATTLIST Name xml:lang CDATA "de">
```

Dieses Beispiel schränkt ein, dass das Wurzelement Menschheit mindestens ein Element mit dem Namen Mensch enthalten muss. Das Element Mensch wiederum kann die beiden Elemente Name und Position genau einmal besitzen, muss diese aber nicht zwangsläufig spezifizieren. Name und Position beinhalten nur für den Computer lesbare Informationen. PCDATA steht für „**P**arsable **C**haracter **D**ATA“ und bedeutet soviel wie „lesbarer textueller Inhalt“. Jedes Name-Attribut besitzt automatisch den Standardwert **xml:lang="de"**, ist somit sprachlich gesehen Deutsch. Mit Hilfe eines **#REQUIRED** am Ende könnten beliebige Attribute verpflichtend gemacht werden.

Demnach ergeben sich folgende Bedeutungen der Symbole +, ? und *

- + Das Element muss **mindestens einmal** vorkommen
- ? Das Element kann **einmal** oder **gar nicht** vorkommen
- * Das Element kann **beliebig oft** oder **gar nicht** vorkommen

Um diese DTD nun in einem XML-Dokument nutzen zu können, kann sie entweder direkt in dem Dokument stehen oder aber per Anweisung aus einer externen Datei ausgelesen werden. Dies ist vor allem bei großen DTDs empfehlenswert. Die direkte Einbindung erfolgt folgendermaßen:

```
<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE Menschheit [
  <!ELEMENT Menschheit (Mensch+)>
  <!ELEMENT Mensch (Name?, Position?)>
  ...
]>
<Menschheit>
  ...
</Menschheit>
```

Ein Verweis für eine separate Datei mit dem Namen menschheit.dtd sieht so aus:

```
<!DOCTYPE Menschheit SYSTEM "menschheit.dtd">
```

Abschließend zum Thema DTD sei darauf hingewiesen, dass „Konstanten“ festgelegt werden können. Genauer gesagt handelt es sich hierbei um sogenannte **Entitäten**, welche einen Namen und einen Wert besitzen. Eine durch die Anweisung

```
<!ENTITY hm "Hans Mustermann">
```

festgelegte Entität kann mit &hm; referenziert werden. In jedem Dokument, welches dieses DTD benutzt, wird anstelle von &hm; demnach „Hans Mustermann“ ersetzt.

DTD gilt mittlerweile als veraltet und soll mittelfristig durch XSD ersetzt werden. Aus Kompatibilitätsgründen kann jedoch jede DTD in eine XSD umgewandelt werden. Auf

Grund der Komplexität von XSD seien hier nur die Vorteile gegenüber DTD erwähnt, eine genaue Beschreibung findet nicht statt.

In XSD kann angegeben werden, wie oft ein Element vorkommen darf. Zudem kann genau festgelegt werden, an welcher Stelle welches Element vorkommen darf. XSD gibt Standarddatentypen wie **integer** oder **string** vor, welche genutzt werden können. So ist es zum Beispiel möglich, dass bestimmte Attribute nur Zahlenwerte beinhalten dürfen. Diese Datentypen können allerdings selbst erweitert bzw. verändert und eigene Datentypen kreiert werden. Bereits vorhandene XSDs können eingebunden und erweitert werden. XSD benutzt zur Beschreibung der Struktur selbst XML, sodass bereits bestehende Parser und/oder Programmteile zur Auswertung genutzt werden können. Dies ist eines der Hauptargumente für XSD.

2.4 Ist XML alleine ausreichend für das Semantic Web?

Nach der vorhergehenden Einführung in XML muss man sich die Frage stellen, ob eine so vielfältige und flexible Sprache nicht ausreicht, um die technischen Anforderungen des Web 2.0 bewerkstelligen zu können. XML alleine ist hierzu nicht in der Lage. Der Grund hierfür ist die Tatsache, dass die Informationen für den Computer unverständlich sind. So kann beispielsweise der Computer nicht wissen, was sich hinter dem Wort Bank verbirgt. Für ihn ist es nichts weiter als eine Aneinanderreihung verschiedener Buchstaben. Woher soll der Computer wissen, ob eine Bank im Sinne eines Kreditinstituts oder einer Sitzmöglichkeit gemeint ist? Es mangelt an einer Semantik, die der Computer verstehen könnte. Auf Informationen, die nicht explizit angegeben sind, kann nicht geschlossen werden. Ein Mensch kann z. B. die Information, dass Äpfel und Birnen Früchte sind selbst erschließen, die Maschine leider nicht. Genau hier setzt RDF an bzw. auch RDFS und OWL (siehe hierzu die **Ausarbeitung von Leonard Kramer** über RDFS und OWL).

3. Ontologien in RDF

Das **Resource Description Framework** ist eine Sprache zur Beschreibung von strukturierten Informationen. Anwendungen sollen durch RDF in der Lage sein, Informationen über das Web auszutauschen, ohne dass hierbei die ursprüngliche Bedeutung verloren geht. Anders als bei HTML ist die korrekte Darstellung zweitrangig. Vielmehr steht die Kombination und Weiterverarbeitung der Informationen im Vordergrund. Aus diesem Grund wird RDF als grundlegendes Datenformat für das Semantic Web gesehen.

Historisch gesehen wurde eine erste Spezifikation von RDF im Jahre 1999 durch das W3C veröffentlicht. Bei dieser Spezifikation wurde der Schwerpunkt allerdings noch eindeutig auf die Darstellung von Metadaten über Web-Ressourcen gelegt.

Hintergedanke hierbei war, beispielsweise eine Aussage über Autor und Lizenzbedingungen einer Webseite tätigen zu können. Eine Erweiterung zur allgemeinen Darstellung semantischer Informationen mit der Vision des Semantic Web fand erst im Laufe der Zeit statt. Im Februar 2004 wurde als Resultat der Überarbeitung eine erweiterte Version publiziert.

Werkzeuge, die einem den Umgang mit RDF Dokumenten erleichtern, sind heute in einer großen Vielzahl verfügbar. Fast jede Programmiersprache bietet mittlerweile diverse Bibliotheken zum Umgang mit RDF an. Selbst kommerzielle Datenbankhersteller⁹ bieten mittlerweile Erweiterungen für ihre Produkte an.

3.1 RDF – Einführung

RDF-Dokumente beschreiben gerichtete Graphen. Diese bestehen aus eindeutig benannten Knoten und gerichteten Kanten („Pfeile“). Hierbei lässt sich schon der erste Unterschied zu XML feststellen: in XML liegt eine Baumstruktur vor. Solch eine hierarchische Struktur ist bei RDF nicht gewollt. Bei dem Satz „Der Mensch stammt ab vom Affen“ lässt sich für „stammt ab vom“ keine logische Unterordnung finden; es handelt sich vielmehr um eine Aussage.

3.2 URI – Uniform Resource Identifier

Ein Problem, das es beim Zusammenfügen von Dokumenten unterschiedlicher Herkunft zu beheben gilt, ist die Vergabe eines eindeutigen Namens. Verschiedene Ressourcen können in unterschiedlichen Dokumenten gleich benannt sein. Um diesem Problem vorzubeugen, werden alle Ressourcen mit einem Namensraum versehen. Aus „stammtAbVon“ wird dann „http://www.stammbaum.org/stammtAbVon“. Dieser Namensraum muss keine gültige Adresse im Sinne einer URL sein, sondern kann auch ganz bewusst fiktiv sein.

URIs beginnen immer mit einem Schema-Bezeichner wie „http“. Hierbei ist unbedeutend, dass http ursprünglich das HyperText Übertragungsprotokoll bezeichnet. Es ist auch nicht wichtig, ob diese URL existiert oder dort relevante Informationen abrufbar sind.

Sowohl Kanten als auch Knoten sind mit URIs beschriftet. Einzige Ausnahme hierbei sind sogenannte „Blank Nodes“, welche in der Regel nur zum Gruppieren genutzt werden¹⁰.

3.3 Literale

Mit URIs wurde die Möglichkeit dargelegt, abstrakte und nicht direkt darstellbare Ressourcen zu beschreiben. Diese Ressourcen können nun durch jede Anwendung anders interpretiert werden. Manchmal ist aber genau das nicht gewünscht, wenn es sich zum Beispiel um für den Menschen eindeutige Zahlenwerte handelt. Genau diese Werte sollen auch für den Computer eindeutig lesbar sein. Diese Datenwerte werden in RDF über **Literale** dargestellt. Der Wert eines Literals wird über eine Zeichenkette beschrieben, welche auf Grund ihres **Datentyps** dann interpretiert werden kann. Die Angabe „039“ beschreibt die gleiche natürliche Zahl wie die Angabe „39“, allerdings handelt es sich hierbei um unterschiedliche Zeichenketten.

Literale werden in RDF-Graphen als Rechtecke dargestellt und können nie selbst als Knoten fungieren. Im Folgenden wird ein Beispiel eines RDF Graphen mit Literalen dargestellt:

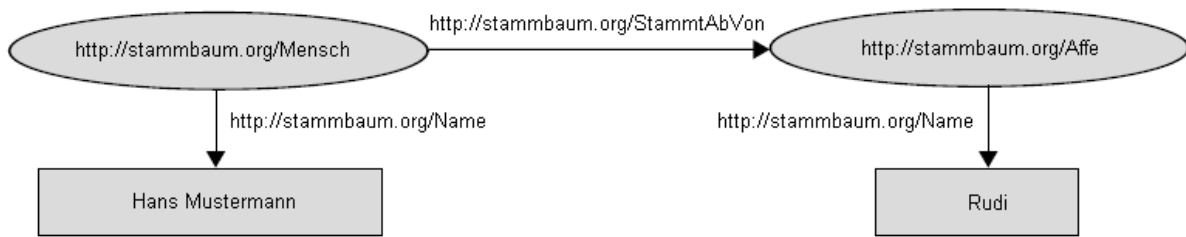


Abbildung 2: RDF-Graph mit Literalen

Diese Abbildung ist für den Menschen leicht verständlich, der Computer kann hiermit allerdings nichts anfangen. Der Graph wird deshalb zunächst in Tripel aufgeteilt und danach in einer bestimmten Syntax abgespeichert.

3.4 Vom Graph zum Tripel

In RDF besitzt jede Kante eines Graphen einen Anfangspunkt, einen Endpunkt und eine eigene Beschriftung. Man spricht hierbei von **Subjekt**, **Prädikat** und **Objekt** – es handelt sich um ein **Tripel**. Mit der Darstellung jeder Kante als Tripel kann der vollständige Graph beschrieben werden. Literale können hierbei ausschließlich Objekt sein, da sie, wie bereits vorher beschrieben, nicht als Ausgangsknotenpunkt für weitere Kanten fungieren können.

Es liegt nun nahe, den kompletten Graphen einfach durch eine Aneinanderreihung von Kanten zu beschreiben und so für den Computer lesbar zu machen. Ursprünglich wurde genau dieses in der von Tim Berners-Lee stammenden **Notation 3** gemacht, welche sogar Regeln und Pfade umfasst. Auf Grund der daraus resultierenden Komplexität wurde 2004 von der W3C eine gekürzte N3 Fassung unter dem Namen **N-Triples** als RDF Syntax empfohlen. N-Triples allerdings wurde später wiederum um einige Kurzschreibweisen erweitert, sodass nun eine inoffizielle **Turtle** (Terse RDF Triple Language) Syntax als de facto Standard vorhanden ist¹¹.

Der Graph aus Abbildung 2 in der Turtle Syntax sieht demnach so aus:

```

<http://stammbaum.org/Mensch>
  <http://stammbaum.org /stammtAbVon> <http://stammbaum.org /Affe> .

<http://stammbaum.org /Mensch>
  <http://stammbaum.org /Name> "Hans Mustermann" .

<http://stammbaum.org /Affe>
  <http://stammbaum.org /Name> "Rudi" .

```

URIs werden von eckigen Klammern umschlossen, während Literale in Anführungszeichen geschrieben werden. Jede Aussage wird mit einem Punkt beendet. Leerzeichen und Zeilenumbrüche wurden nur zur besseren Lesbarkeit eingefügt und spielen bei der Interpretation keine Rolle.

Bei XML besteht die Möglichkeit, Namensräume zu deklarieren. Turtle nutzt dieses Feature, sodass URIs nun mit einem Präfix abgekürzt werden:

```

@prefix      sb:    <http://stammbaum.org/> .
sb:Mensch    sb:stammtAbVon    sb:Affe .
sb:Mensch    sb:Name            "Hans Mustermann" .
sb:Affe      sb:Name            "Rudi" .

```

Hierbei sei erwähnt, dass die Bezeichnung der Form „Präfix:Name“ auch **qualified name** genannt wird.

RDF Beschreibungen enthalten oftmals Tripel, welche sich auf das gleiche Subjekt beziehen. Hierfür kann folgende **kürzere Schreibweise** verwendet werden:

```

@prefix      sb:    <http://stammbaum.org/> .
sb:Mensch    sb:stammtAbVon    sb:Affe ;
              sb:Name            "Hans Mustermann", "Hansi" .

```

Die Trennung durch das Semikolon bewirkt, dass das nachfolgende Tripel sich auf das gleiche Subjekt bezieht. Unterschiedliche Objekte mit dem gleichen Subjekt und Prädikat können durch ein Komma getrennt beschrieben werden. Der Graph besteht in diesem Fall aus zwei Knoten, drei Kanten und zwei Literalen.

3.5 Serialisierung in XML

Die Schreibweise in der Turtle Syntax ist für den Menschen relativ gut lesbar und für den Computer eindeutig zu interpretieren. Dennoch gibt es kaum Bibliotheken, welche die Turtle Syntax in dieser Art und Weise einlesen können. Anders ist dies bei XML. Wie bereits vorher beschrieben bietet praktisch jede Programmiersprache eine Bibliothek zum Umgang mit XML-Dokumenten. Aus diesem Grund ist die wichtigste Darstellung von RDF eine XML-Schreibweise. Dass XML eine Baumstruktur darstellt und keinen Graphen, ist hierbei kein Hindernis, da nur die syntaktische Struktur durch XML vorgegeben wird. Die Umsetzung der Tripel in XML muss allerdings auch hierarchisch erfolgen. Als Grundlage wird die kürzere Schreibweise der Turtle Syntax verwendet, wobei nach dem Subjekt eines Tripels gruppiert wird. Die in 3.4 beschriebenen Regeln sehen demnach in XML folgendermaßen aus:

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:sb="http://stammbaum.org">
  <rdf:Description rdf:about="http://stammbaum.org/Mensch">
    <sb:stammtAbVon>
      <rdf:Description rdf:about="http://stammbaum.org/Affe">
        </rdf:Description>
      </sb:stammtAbVon>
    </rdf:Description>
  </rdf:RDF>
```

Zunächst findet sich in der ersten Zeile die XML Deklaration mit der optionalen Angabe von Version (1.0) und Zeichenkodierung (Unicode). Anschließend folgt das Wurzelement des RDF Dokumentes **rdf:RDF**. Im Wurzelement werden die Namensräume **rdf** und **sb** durch **xmlns:rdf** und **xmlns:sb** deklariert. Der Namensraum **rdf** kann zwar beliebig gewählt werden, die Nutzung der Standardbezeichnung sollte aber beibehalten werden. Alle nachfolgenden Elemente, welche in RDF eine besondere Bedeutung haben, sind demnach an diesem Namensraum erkennbar. Innerhalb unseres Wurzelementes **rdf:RDF** befindet sich unser Tripel „Mensch stammtAbVon Affe“. Subjekt und Objekt werden jeweils durch das Element **rdf:Description** beschrieben, wobei jeweils durch das Attribut **rdf:about** eine eindeutige Bezeichnung vergeben wird. Das Prädikat wird direkt angegeben, wie am Element **sb:stammtAbVon** zu sehen ist. Anders als die

Elemente von Subjekt und Objekt ist der Namensraum des Prädikats im Tag angegeben.

```
<rdf:Description rdf:about="http://stammbaum.org/Mensch">
  <sb:Name>Hans Mustermann</sb:Name>
  <sb:stammtAbVon>
    <rdf:Description rdf:about="http://stammbaum.org/Affe">
      <sb:Name>Rudi</sb:Name>
    </rdf:Description>
  </sb:stammtAbVon>
</rdf:Description>
```

Dies kann jedoch noch weiter verkürzt werden, nämlich indem Elementbeschreibungen als Attribute angegeben werden.

```
<rdf:Description rdf:about="http://stammbaum.org/Mensch"
  sb:Name="Hans Mustermann">
  <sb:stammtAbVon rdf:resource="http://stammbaum.org/Affe" />
</rdf:Description>
<rdf:Description rdf:about="http://stammbaum.org/Affe" sb:Name="Rudi" />
```

Die Verwendung von **rdf:resource** bezieht sich in diesem Beispiel direkt auf das Objekt des Tripels, sodass kein erneutes `rdf:Description` Element mehr von Nöten ist, um das Objekt zu beschreiben.

Auffällig ist, dass trotz der Festlegung des Namensraumes `xmlns:sb="http://stammbaum.org/"` trotzdem nicht das Kürzel `sb:` verwendet wird. Dies hängt damit zusammen, dass der Doppelpunkt in Werten von XML Attributen **falsch interpretiert** würde, nämlich als Teil einer URI. Damit dieses Problem umgangen werden kann, existieren zwei Möglichkeiten. Zunächst lässt sich im Wurzelement mit dem Attribut **xml:base** " ein Standardnamensraum festlegen, der für alle Elemente gilt.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xml:base="http://stammbaum.org">
```

Dadurch fällt die zusätzliche Angabe in Attributwerten weg.

Alternativ kann eine Entität mit

```
<!ENTITY sb "http://stammbaum.org">
```

deklariert werden, sodass die Verwendung von "&sb;Mensch" zulässig wird.

3.6 Der W3C RDF Validator

Wie bereits erwähnt, können XML-Dokumente auf ihre Gültigkeit in Bezug auf Wohlgeformtheit und DTD Validität hin überprüft werden. Eine solche Validierungssoftware wird auch für RDF durch das W3C bereitgestellt. In einem Online-Formular kann entweder der Text direkt oder eine bestimmte URL eingefügt und das Dokument anschließend validiert werden. Selbst die Zeichnung des Graphen wird unterstützt.

Der Validator ist unter "<http://www.w3c.org/RDF/Validator>" zu erreichen. Im Anhang befindet sich hiervon ein Screenshot.

3.7 Vokabular

In diesem Kapitel wurden nun grundlegende Funktionen und Eigenschaften von RDF erläutert und URIs zur Beschreibung von Ressourcen eingeführt. Es besteht allerdings immer noch das Problem, dass gleiche Ressourcen in verschiedenen Dokumenten anders benannt werden können. Für diesen Zweck gibt es **wohldefinierte Vokabulare**. Als Vokabular bezeichnet man eine Zusammenstellung aus festgelegten Bezeichnern. Ohne es zu wissen, wurde bereits das festgelegte Vokabular für RDF¹³ selbst benutzt. Das **rdf:Description** Element beispielsweise stammt aus dem RDF Vokabular und hat als volle URI <http://www.w3.org/1999/02/22-rdf-syntax-ns#Description>. Es besitzt hierdurch eine allgemein anerkannte Bedeutung.

4. Zusammenfassung und abschließende Bemerkung

Die mittlerweile beachtlich große Menge an Informationen im Web stellt uns vor neue Probleme wie zum Beispiel das Auffinden relevanter Informationen. Mit Hilfe der Technologien, die im Semantic Web zur Anwendung kommen, soll uns genau dies ermöglicht werden. Aus technischer Sicht wird hierfür eine Sprache benötigt, die jeder Computer verstehen und problemlos kommuniziert werden kann. Als Standard hat sich hier die Sprache XML etabliert, welche die Grundlage für andere beschreibende Sprachen darstellt. Wie wir festgestellt haben, basiert auch RDF auf XML. RDF besitzt eine Graphenstruktur, welche durch Tripel (Subjekt, Prädikat, Objekt) realisiert wird. Eine anschließende Serialisierung der Tripel in XML ermöglicht das Speichern und Auslesen dieser Informationen. Beziehungen zwischen einzelnen Ressourcen können durch RDF problemlos dargestellt werden, komplexe Zusammenhänge wie Hierarchien erfordern allerdings weitere Technologien wie RDFS und OWL, die im Rahmen dieser Ausarbeitung nicht weiter ausgeführt werden.

Die Forschung im Bereich der künstlichen Intelligenz, wozu man das Semantic Web zweifelsfrei zählen kann, steht trotz aller Bemühungen und Technologien noch ganz am Anfang. Bis der Computer komplexe Zusammenhänge selbst erschließen und dem Menschen neues Wissen offenbaren kann, werden noch viele Jahrzehnte vergehen. Mit immer neuen Erfindungen können wir uns aber jetzt schon den Umgang mit dem Internet erleichtern. Nur so ist es dem Medium Internet möglich, ein Massenmedium zu werden und das Nutzerspektrum zu erweitern.

5. Abbildungsverzeichnis

Abbildung 1: Offizielles W3C Logo	1
Abbildung 2: RDF-Graph mit Literalen.....	13

6. Quellenverzeichnis

1. **IT Handbuch IT Systemelektroniker/-in Fachinformatiker/-in**
Westermann Schulbuchverlag GmbH
4. Auflage 2005, Seite 229
2. **Semantic Web: Grundlagen**
Springer Verlag
1. Auflage, Kapitel 2 bis 4
3. **Wikipedia: XML – eXtensible Markup Language**
http://de.wikipedia.org/wiki/Extensible_Markup_Language
Stand 15. Juli 2008
4. **W3C Recommendation**
<http://www.w3.org/TR/2006/REC-xml11-20060816/>
Stand 15. Juli 2008
5. **Wikipedia: DTD – Document Type Definition**
http://de.wikipedia.org/wiki/Dokumenttypdefinition#DTD_in_XML
Stand 15. Juli 2008
6. **XHTML Dokumenttypendefinition der W3C**
<http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>
Stand 15. Juli 2008
7. **DTD: Regeln für XML-Dokumente**
<http://www.tecchannel.de/webtechnik/entwicklung/401511/>
Stand 15. Juli 2008
8. **Wikipedia: RDF – Resource Description Framework**
http://de.wikipedia.org/wiki/Resource_Description_Framework
Stand 15. Juli 2008
9. **Oracle Semantic Technologies Center**
http://www.oracle.com/technology/tech/semantic_technologies/index.html
Stand 08.08.2008

10. **Microsoft Developers Network: What is an RDF Blank Node?**
<http://msdn.microsoft.com/en-us/library/aa303696.aspx>
Stand 09.08.2008
11. **Turtle – Terse RDF Triple Language**
<http://www.dajobe.org/2004/01/turtle/>
Stand 09.08.2008
12. **Offizielles W3C Logo zum Semantic Web**
<http://www.w3.org/2007/10/sw-logos.html>
Stand 11.08.2008
13. **RDF Vokabular**
<http://www.w3.org/1999/02/22-rdf-syntax-ns>
Stand 12.08.2008

7. Anhang

7.1 Screenshot des W3C Validators

W3C RDF Validation Results - Microsoft Internet Explorer

Adresse: <http://www.w3.org/RDF/Validator/ARPServlet>

W3C[®] RDF[®] Validation Service

Home Documentation Feedback

Validation Results

Your RDF document validated successfully.

Jump To:

- Source
- Triples
- Messages
- Graph
- Feedback
- Back to Validator Input

Triples of the Data Model

Number	Subject	Predicate	Object
1	http://stammbaum.org/Mensch	http://stammbaum.org/Name	"Hans Mustermann"
2	http://stammbaum.org/Mensch	http://stammbaum.org/stammtAbVon	http://stammbaum.org/Affe
3	http://stammbaum.org/Affe	http://stammbaum.org/Name	"Rudi"

The original RDF/XML document

```
1: <?xml version="1.0"?>
2: <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:sb="http://stammbaum.org/">
3:
4:   <rdf:Description rdf:about="http://stammbaum.org/Mensch" sb:Name="Hans Mustermann">
5:     <sb:stammtAbVon rdf:resource="http://stammbaum.org/Affe" />
6:   </rdf:Description>
7:   <rdf:Description rdf:about="http://stammbaum.org/Affe" sb:Name="Rudi" />
8:
9: </rdf:RDF>
10:
```

Graph of the data model

```
graph LR
  M([http://stammbaum.org/Mensch]) -- "http://stammbaum.org/Name" --> HM[Hans Mustermann]
  M -- "http://stammbaum.org/stammtAbVon" --> A([http://stammbaum.org/Affe])
  A -- "http://stammbaum.org/Name" --> R[Rudi]
```