Präsentation Schweizerische Bundeskanzlei, 13. November 2001

# *Entwicklung von Schemata für den XML-Dokumentenaustausch*

## mit Hilfe des Resource Description Framework (RDF)

John McClure
jmcclure@hypergrove.com




**Einführung.** Eine Folge der Verwendung der Extensible Markup Language (XML)-Technologie ist ein neuer Fokus auf Textdokumente, die zwischen Organisationen und Personen ausgetauscht werden. In der Vergangenheit konzentrierten sich Software-Entwickler auf binäre Nachrichten, die zwischen Anwendungen ausgetauscht wurden, welche auf Client- und Server-Computern ausgeführt wurden. Dabei treten zwei grundlegende Fragen bezüglich der Architektur auf: (1) Welche Informationen in einem Dokument müssen von XML-Elementen separat mit Tags versehen werden, damit sie extrahiert und dann verwendet werden können. (2) Was sind die optimalen Formate für und Anordnungen der XML-Elemente im Dokument. Die Antwort auf die erste Frage wird durch ein **Dokumentschema** dargestellt, das angibt, was in einem Dokument mit Tags versehen wird. Die Antwort auf die zweite Frage – wie Dokument-Instances mit XML-Elementen kodiert werden – wird teilweise durch die zur Aufzeichnung eines Dokumentschemas verwendeten Technologien bestimmt.

Es gibt zwei XML-Dialekte zur Darstellung des Dokumentschemas : (1) *RDF-Schema* und (2) *XML-Schema*. Ein Thema heute ist der Unterschied zwischen diesen Dialekten, die beide wichtige technische Empfehlungen des Worldwide Web Consortium (W3C) darstellen, d. h. sie haben die größtmögliche Unterstützung durch das W3C. Das Ziel dieser Präsentation besteht darin, Ihnen zu helfen, informierte Entscheidungen über die für Ihre Organisationen und Software-Anwendungen am besten geeigneten Technologien zu treffen. Ich möchte Ihnen dazu die Architektur unserer Standards näher bringen, die die beiden Schemadialekte auf eine Art und Weise kombiniert, die ihre besten Eigenschaften optimal nutzt.

**Vortragender.** Ich heiße John McClure, bin der Gründer von Hypergrove Engineering, einer Firma, deren Ziel die Entwicklung von Software ist, die bei der Formatierung und Verwaltung von Miet- und Kaufverträgen eingesetzt wird. Ich möchte offene, öffentliche Standards für juristische Informationen schaffen und letztendlich der Architekt für das Data Consortium werden, einer gemeinnützigen Vereinigung innerhalb des Immobiliensektors in den USA. Ich leite zwei Arbeitsgruppen innerhalb von LegalXML, einer Vereinigung von Gerichten, Polizeibehörden und Herausgebern von juristischen Schriften. Ich leite die Vertragsarbeitsgruppe von LegalXML und zusammen mit Murk Muller die Wörterbucharbeitsgruppe, in der wir ein mit dem Resource Description Framework kompatibles Wörterbuch schaffen. Ich komme aus dem Wirtschaftssektor, war aber viel im EDV-Bereich tätig. Bei IBM war ich Entwickler der Desktop-Umgebung des OS/2-Betriebssystems und später Leiter eines Teams, das einen SGML-Dokumenteditor entwickelte. Bei MCI war ich für Anwendungs- und Datenarchitekturen verantwortlich. In dieser Position ermutigte ich stets zur firmenweiten Verwendung der Extensible Markup Language.

**Links.** http://www.w3.org, http://www.dataconsortium.org, http: //www.legalxml.org, http://www.hypergrove.com

## Metadaten-Spezifikation

| *Merkmal* | *RDF-Schema* | *XML-Schema* |
|---|---|---|
| Allgemeiner Name | "Wörterbuch"(Dictionary), "Ontologie" | "Dokumentschema" |
| ➡ Allgemeine Ausrichtung | Definiert Namen für Dinge<br>Definiert Objektmodelle | Definiert XML-Elemente<br>Definiert ein Transportprotokoll |
| Grundlegende Elemente | `<Class>`<br>`<Property>` | `<Element>`<br>`<Attribute>` |
| Grundl. Datenarten | XML-Schema-Datentypen | XML-Schema-Datentypen |
| Beschränkungsunterst. | Ja | Ja |
| Inheritance-Unterstütz. | mehrfache Vererbung | einfache Inheritance |
| Synonymunterstützung | `<sameClassAs>`<br>`<samePropertyAs>` | ? |

Das Ziel jedes Schemas ist, Metadaten so zu definieren, dass sie für Anwendungssoftware nützlich sind. Metadaten umfassen Dokumente innerhalb der Software, Felder in den Dokumenten und zulässige Werte für Felder. RDF-Schema (RDFS) und XML-Schema (XMLS) sind die XML-Dialekte für die Metadaten-Spezifizierung, die das W3C empfiehlt, obgleich es auch andere gibt (ASN, DDL, DTD, Relax NG, XMI). RDFS-Repositories werden oft als Wörterbücher oder in der Fachsprache auch als semantische Ontologien bezeichnet. XMLS entstand aus Document Type Definitions (DTDs), seine Repositories heißen dementsprechend „Dokumentschemata".

Der Hauptunterschied zwischen den beiden Dialekten besteht darin, dass **RDFS** Namen für **Informationselemente** definiert (dabei werden „Class"-Elemente verwendet), während **XMLS** Namen für **XML-Elemente** definiert (und dabei „Element"-Elemente verwendet). Das ist wichtig. Weil RDFS Typen von Programmobjekten definiert, die von der Software manipuliert werden, und XMLS ein spezifisches Transportprotokoll für das Lesen und Schreiben der Daten durch die Software verwendet, können RDFS-Wörterbücher auf mehrere Transportprotokolle angewendet werden; bei XMLS-Schemata ist das nicht der Fall. Ein RDFS-Wörterbuch ist auf eine XML-Darstellung (fungiert als XML-Schema), eine HTML/SVG/XFORMS-Darstellung oder selbst eine binäre Darstellung anwendbar. Das XML-Schema ist dazu nicht in der Lage.

Damit RDFS-Metadaten funktionell XMLS-Metadaten entsprechen, muss tiefer geforscht werden. Beide bieten vertraute Inhaltsmodelle, Beschränkungsspezifikationen und einen identischen Satz intrinsischer Datentypen (Zeichenfolgen, Zahlen, Daten, URLs usw.). Wie die meisten objektorientierten Programmiersprachen, ist mit RDFS jedoch eine Mehrfach-Inheritance möglich, mit XMLS nicht. Darüber hinaus bietet RDFS Synonym- und Antonymunterstützung, Konzepte, die XMLS fremd sind. RDFS kann wie XMLS die Grundfunktionen der Document Type Definitions durchführen, der Metadaten-Funktion, die sie jeweils ersetzen sollten – Instance-Datenstromvalidierung.

Da das Metadaten-Repository einer Organisation meist eine große Datenmenge enthält, muss schließlich gefragt werden: Ist es eine gute Investition der Organisation, zwei Metadaten-Repositories zu erstellen und zu pflegen? Die kurze Antwort lautet ja, wenn das XMLS-Repository bewusst klein ist, werden die meisten Metadaten im RDFS-Repository gespeichert. Bei der längeren Antwort spielt die Notwendigkeit von Datenstromtransformationen und Abfrageanforderungen eine Rolle.

## *Instance-Dokumentkodierung*

| ***Merkmal*** | ***RDF-Schema*** | ***XML-Schema*** |
|---|---|---|
| Laufzeitunterstützung | Teil der Architektur | Nicht angesprochen |
| Ressourcen-/Objekttyp | Dynamisch | Statisch |
| Beständige Identifikatoren (Zuweisung/Referenz) | `rdf:ID` – für neue Ressourcen<br>`rdf:about` – für neue Daten<br>`rdf:resource` – Referenzen | Nicht angesprochen |
| XML-Kodierungsstile | Attribute = Elemente | Nicht flexibel |
| Meta-Aussagen | Ja | N. angesprochen |
| Elementmuster | Substantiv–Verb–Substantiv | N. angesprochen |

Das Resource Description Framework bietet mehr als nur ein Metadaten-Repository. Seine Architektur umfasst eine Laufzeitumgebung, die RDF-Datenströme als Reihe von Aussagen identifiziert, die aus einem Subjekt, Prädikat und Objekt (sog. Tuple) bestehen. RDF bietet auch Meta-Aussagen, d. h. eine Aussage, deren Objekt eine andere Aussage ist (z. B.: „Hans sagte, *die meisten Metadaten müssten in einem RDFS-Wörterbuch gepflegt werden*"). Die Laufzeitfunktion ermöglicht daher, dass eine Anwendung anhand der Werte zugewiesener Prädikate und Objekte wie bei relationalen Datenbankauswahlen alle Subjekte im Dokument „auswählt".

Während der Datenbankentwicklung und -programmierung treten Probleme auf, wenn eine Einheit mehrere Typen hat und die zugrunde liegende Technologie nur zulässt, dass der Einheit ein einziger Typ zugewiesen wird. Im RDF wird dieses Problem behoben, weil einer Ressource mehrere Typen zugewiesen werden können. Der herkömmliche Ansatz zur Angabe, dass etwas mehrere Typen hat, besteht darin, boolesche Elemente (Flags) zu verwenden; im RDF nimmt die Ressource automatisch die Attribute (oder Eigenschaften), die ihren Typen zugewiesen sind, an, wodurch die zu modellierenden Realitäten elegant in Übereinstimmung gebracht werden (siehe Beispiel unten).

RDF umfasst auch einen Mechanismus, der angibt, wenn ein XML-Element die Definition einer neuen Ressource darstellt (Attribut `rdf:ID` wird verwendet) oder Informationen über eine vorhandene Ressource enthält (Attribut `rdf:about` wird verwendet). RDF spricht auch Referenzen einer vorhandenen Ressource an (`rdf:resource` wird verwendet).

RDF ermöglicht, dass mehrere Kodierungen die gleichen Bedeutungen haben. Das Element `Anwaltskanzlei` unten könnte z. B. als `<Anwaltskanzlei dc:Title='Schmitt &amp; Jahn'/>` definiert werden. Die RDF-Laufzeit legt die Information `dc:Title` als Attribut (oder Eigenschaft) eines Ressourcenobjekts `Anwaltskanzlei` fest, ob sie nun als XML-Attribut oder XML-Element kodiert wurde.

*Beispiel für RDF-Markup*

```
<Zeuge rdf:ID='http://www.uscourts.gov/USA.FederalCourt.CourtCase.Witness#AB1234567890'>
   <rdf:type rdf:resource='#Attorney'/>
   <repräsentiert rdf:resource='http://www.uscourts.gov/USA.FederalCourt.CourtCase.Defendant#FOO1234890'/>
   <angestelltVon>
     <Anwaltskanzlei rdf:about='http://www.dot.net/USA.Business#AB1234567890'>
        <dc:Title>Schmitt &amp; Jahn</dc:Title>
     </Anwaltskanzlei>
   </angestelltVon>
</Zeuge>
```

## *Strategie für einen Standard*

***Fakt***: Die meisten Dokumente u. Formulare werden heute in HTML und PDF kodiert
***Fakt***: SVG ersetzt PDF; XFORM ersetzt HTML-Formulare
***Fakt:*** XHTML, SVG und XFORM sind XML-Dialekte

***Rezept für einen revolutionären Standard***
Anforderung, dass Verleger Nichtpräsentations-„Dokumente" austauschen
Anforderung, dass Verleger und alle Kunden neue Tools/Browser kaufen/erlernen
Anforderung, dass Verleger redundante XML-Elemente in Datenströmen erstellen

***Rezept für einen evolutionären Standard***
Änderungen der aktuellen Praktiken und der aktuellen Tools minimieren
HTML/SVG/XFORM-Elemente anmerken: ihre Attribute *name* verwenden
Keine (oder wenige) neue Elemente müssen zu aktuellen Datenströmen hinzugefügt werden

Das übliche Ziel von Organisationen, die XML-Standards für eine vertikale Industrie einführen, besteht darin, die Dokumente, die innerhalb der vertikalen Industrie erstellt werden, auszutauschen. Ein „Dokument" ist Präesentationsmaterial, das bis jetzt mit Hilfe von HTML oder Adobes PDF-Sprache dargestellt wurde, die beide nicht mit XML konform sind und fraglos durch XML-kompatible Dialekte ersetzt werden: XHTML und XFORM ersetzen HTML, die Scalable Vector Graphics (SVG)-Sprache ersetzt PDF. Dabei werden wohl alle Dokumente bald von Publishern mit Hilfe von XHTML, XFORM und SVG kodiert werden.

Ist es daher nicht ironisch, dass Konsortien, die Standards erstellen, Nicht-Präsentationsdialekte von XML erstellen, die für den Austausch von Präsentationsmaterial verwendet werden? Definitionsgemäß scheinen Konsortien Standards für den Austausch von Datenbanken, nicht Dokumenten, zu erstellen, da die Informationen nicht ihre endgültige Form angenommen haben. Publisher, die dem Austausch verpflichtet sind, müssen daher sowohl Präsentationsmaterial erstellen als auch eine Datenbank, die das Dokument darstellt. Das ist nicht akzeptabel, da die letztendliche Auswirkung des Standards aktuelle Geschäftspraktiken, die (immer mehr) auf XHTML, SVG und XFORM beruhen, maximal unterbrechen oder stören. Je mehr Störungen und Unterbrechungen, desto weniger wahrscheinlich fasst der vorgeschlagene Standard in Anteilseignerorganisationen Fuß.

Die HTML-, SVG- und XFORM-Dialekte von XML müssen als Fundament für den Dokumentenaustausch eingesetzt werden. Zum Glück umfassen diese drei Dialekte bereits die Mechanik, die Standardorganisationen dazu benötigen. Die meisten ihrer Elemente haben ein Attribut *name*, in den Publisher einen ***strukturierten gepunkteten Namen*** einschließen können. Dieser gepunktete Name wird zur Etikettierung der vom Element dargestellten Informationen eingesetzt.

Anders gesagt müssen sich Standardkonsortien auf die Etablierung der Struktur von gepunkteten Namen konzentrieren, nicht auf die Einführung völlig neuer XML-Dialekte, die hinsichtlich der aktuellen Geschäftspraktiken, Tools und Fertigkeiten und hinsichtlich der Objekte, die dargestellt werden (d. h. Präsentationsdokumente), nicht zusammenhängen.

## *Strukturierte gepunktete Namen*

***In einem HTML-Datenstrom verwendet***

```
<input type='text' name='Zeuge.Vorname' value='Hans'/>
```

***In einem XML-Datenstrom***

```
<Zeuge.Vorname>Hans</Zeuge.Vorname>
```

***In verschachtelte XML-Elemente konvertiert***

```
<Zeuge>
       <Vorname>Hans</Vorname>
</Zeuge>
```

***In einen RDF-Datenstrom konvertiert***

```
<Zeuge>
   <heisst>
        <Vorname>Hans</Vorname>
   </heisst>
</Zeuge>
```

***In ECMA-Software verwendet***

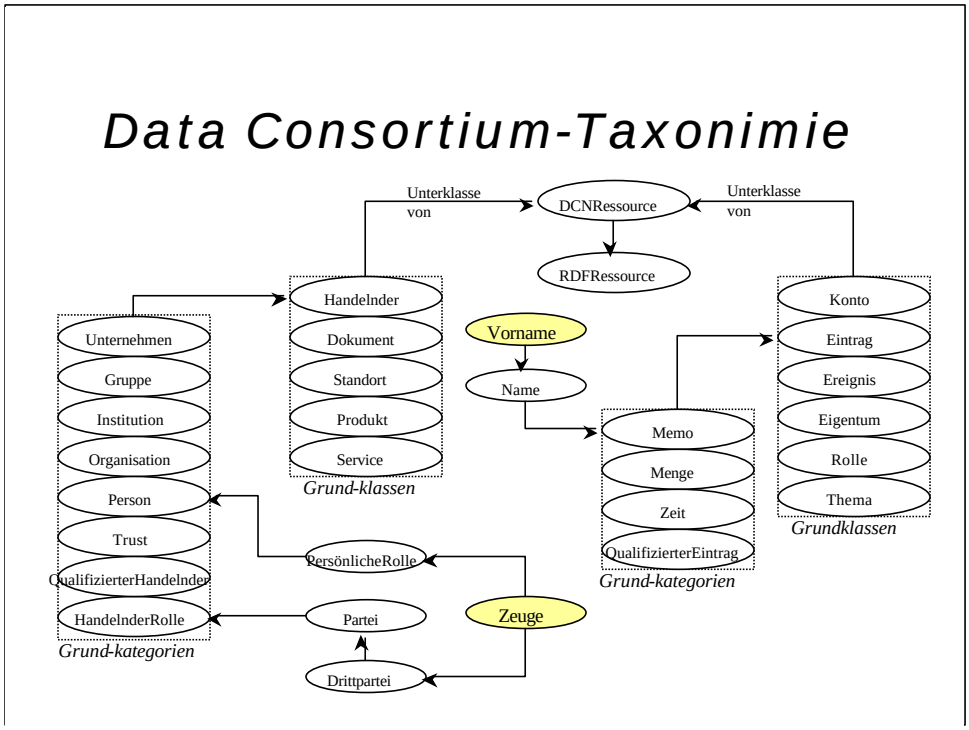
```
Zeuge.Vorname = 'Hans";
```

Wenn in einem HTML-Datenstrom verwendet, wird ein gepunkteter Name naturgemäß zur Bezeichnung des Werts des Attributs *value* in einem `<input> Element verwendet`. Bitte beachten Sie, dass bei Speichern des HTML-Formulars auf einem HTTP-Server der Wert der Attribute *name* und *value* automatisch extrahiert wird, wodurch das gleiche Ergebnis erzielt wird, wie wenn der Wert in einem XML-Datenstrom verwendet wird (zweites Szenario). Die Verwendung eines gepunkteten Namens als Wert des Attributs *name* in einem HTML-Dokument hat jedoch auch einen Nachteil. Dynamische HTML (Javascript) ermöglicht eine nahtlose Verwendung des Attributs *name* als Teil eines Eigenschaftsnamens, das bedeutet, dass Javascript-Programme diese (etwas merkwürdige) Funktion nicht verwenden könnten. Wird dies nicht erwünscht, könnte stattdessen ein Namespace-qualifiziertes Attribut verwendet werden. Bei Verwendung des Namespace von Data Consortium, würde das Attribut *dcn:name* „Zeuge.Vorname"' enthalten, während das Attribut *name* z. B. „ZeugeVorname" enthalten würde.

Es ist syntaktisch nützlich, dass der Name eines XML-Element ein gepunkteter Name ist. Bitte beachten Sie jedoch, dass bei Verwendung als Name eines XML-Elements eine Elementverschachtelung (siehe drittes Szenario) nicht möglich ist – was den Vorteil hat, dass der XML-Datenstrom selbst für Menschen verständlicher und einfacheres Debugging wiedergegeben werden kann. Es spricht eigentlich wenig dagegen, benachbarte XML-Elemente, die keine Attributwerte oder keinen Textinhalt aufweisen, standardmäßig nicht ***zufällig*** mit dem Dot-Operator zu kombinieren – die verschachtelte Elementstruktur kann leicht eingefügt oder mechanisch entfernt werden. Eine solche Vorverarbeitung wäre notwendig, um (a) den Datenstrom mit einem DTD-, einem XML- oder einem RDF-Schema zu validieren und (b) den Datenstrom mit Hilfe der Extensible Stylesheet Language (XSL-T) zu verarbeiten oder anderweitig umzuformen.

Im nächsten Szenario wird der gepunktete Name in eine RDF-Darstellung umgewandelt. Das Prädikat („heisst") wird zwischen die Substantive „Zeuge" und „Vorname" anhand der Metadaten im Repository eingefügt, das die Beziehungen zwischen Substantiven identifiziert. Der gepunktete Name „Zeuge heisst Vorname" ist auch gültig, doch ist dieser Name potenziell künstlich und könnte, falls es sich um eine Publisher-Anforderung handelt, Widerstand von Anteilseignern hervorrufen, die einen einfachen, nicht intrusiven Standard für den Informationsaustausch vorziehen.

Der gleiche gepunktete Name kann in einem ECMA Script-Programm (Javascript) verwendet werden. Dem Eigenschafts-Slot „Vorname" des Objekts „Zeuge" wird ein Zeichenfolgewert zugewiesen. Daher schaffen gepunktete Namen Beständigkeit über mehrere heute mit dem Internet verbundene Technologien hinweg. Dazu ist das XML-Schema, das zur Definition von XML-Elementen verwendet wird, nicht geeignet. Das RDF-Schema zur Definition von Objektklassen und ihrer Eigenschaften ist für Repositories, die Operationsmetadaten enthalten, sehr gut geeignet.




Das Wörterbuch des Data Consortium enthält eine Taxonomie für fast 8000 Begriffe im Zusammenhang mit kommerziellen Immobilien in den USA. Ein Teil davon wird in diesem Diagramm dargestellt. Eine *Taxonomie* ist in diesem Kontext eine Anordnung von Begriffen in übergeordneten/untergeordneten Klassenbeziehungen. **DCNResource** ist z. B. eine Unterklasse der RDF-Klasse **Ressource ,** einer Klasse, die ein generisches Objekt im Resource Description Framework darstellt. Eine übergeordnete Klasse könnte als „Kategorie " bezeichnet werden, doch ist sie mehr als nur ein Behälter für andere Begriffe (jeder Objektklasse können Eigenschaften zugewiesen werden, und alle einer übergeordneten Klasse zugewiesenen Eigenschaften werden an deren Unterklassen „vererbt"). Man kann auch sagen, dass Unterklassen Untertypen (das Element `rdf:type` ist daher korrekt bezeichnet) sind. Die Pfeile im Diagramm zeigten auf die übergeordnete Klasse eines Begriffs.

Im Wörterbuch ist ein **Vorname** ein Typ eines **Namens**, der ein Typ des Textmemorandums (**Memo**) ist, das über eine Ressource aufgezeichnet werden kann. Ein **Memo** ist ein Untertyp eines **Eintrags**, einer der elf Grundklassen in der Taxonomie. Die Klasse **Memo** wird als Grundkategorie bezeichnet, weil sie ein unmittelbarer Untertyp einer Grundklasse ist. Weitere Grundkategorieklassen innerhalb von **Eintrag** sind **Menge** (für numerische Datenelemente), **Zeit** (für Uhrzeit- und Datumsdatenelemente) und **Qualifizierter Eintrag** (für Namen, denen ein Modifikatorbegriff als Präfix vorangeht). Die Taxonomie für einen **Zeugen** ist komplexer: es ist eine Unterklasse von sowohl **Drittpartei** als auch **PersonönlicheRolle .** Eine **Drittpartei** ist eine Unterklasse der **Partei,** ein Untertyp von **HandelnderRolle** , dessen übergeordnete Klasse die Klasse **Handelnder** ist**.** Eine **PersönlicheRolle** ist eine Unterklasse einer **Person** sowie eine Unterklasse der Klasse **Handelnder**.

Das Ziel einer Taxonomie ist, Definition und Durchsetzung von Geschäftsregeln bezüglich des Dokumentinhalts zu erleichtern sowie die notwendige Programmierung zu minimieren, während eine Taxonomie verbessert oder korrigiert wird (was für die Entwicklung von Standards grundlegend ist). Weil ein **Zeuge** z. B. ein Untertyp von **Person** ist, kann er wie jedes Objekt **Person** einen **Vornamen** haben. Überdies kann jede der Unterklassen eines Schemas verwendet werden, wenn das Schema die Verwendung eines **Zeugen** zulässt. Eine Klasse **Gegenzeuge** könnte z. B. in die Taxonomie als Unterklasse von **Zeuge** eingefügt und ohne Änderung des Großteils der vorhandenen Software verwendet werden.

Moderne Programmiersysteme erfordern Mehrfachklassen-Inheritance, um die von ihnen manipulierten Daten korrekt zu modellieren. Weil das RDF-Schema Mehrfach-Inheritance zulässt (was im XML-Schema nicht immer möglich ist), stellt es eine bessere Grundlage für ein (verteiltes) Metadaten-Repository als das XML-Schema dar, welches sowohl normative Metadaten als auch organisationsspezifische Metadaten enthält.

## *Mehrfach-Schemadefinitionen*

**DTD** Elemente für Grundklassen und Datentypen
**XML-Schema** Elemente für Grundkategorieklassen (siehe *xsi:type*)
**RDF-Schema** Elemente für spezifische Begriffe (siehe Attribut *name*)

```
<Handelnder xsi:type='Person'  name='Mann'>
<Dokument xsi:type='Gerichtsdokument' name='Schriftsatz'>
<Eintrag  xsi:type='Memo'      name='Vorname'>
<Ereignis xsi:type='Übergeben' name='vertraglich'>
<Standort xsi:type='Anlage'   name='Gericht'>
<Produkt  xsi:type='Möbel' name='Stuhl'>
<Eigentum xsi:type='Immobilien' name='Wohnung'>
<Service  xsi:type='ÖffentlicheVerwaltung' name='Brandschutz'>
<Thema    xsi:type='Finanzen'   name='GeschäftFinanzen'>
```

**DTD** allgemeines Inhaltsmodell für Grundklassen
**XML-Schema** Inhaltsmodell für Grundkategorien + Übersetz. **RDF-Schema** detaillierte Inhaltsmodelle + Übersetzungen

Die Hauptfunktion von Dokumentschemata besteht darin, die Validierung des Inhalts von XML-Dokumenten zu unterstützen. Das Hauptziel der Validierung besteht darin, dass ein Dokument XML-Elemente enthält, die in Stylesheets genannt werden und andere Software später das Dokument verarbeitet. Unser Ansatz schafft mehrere Stufen der Validierung je nach dem verwendeten Schema. Die erste Stufe untersucht das Dokument syntaktisch, um zu gewährleisten, dass es gegen ein DTD- oder XML-Schema validiert, das Definitionen weit definierter Datentypen und Inhaltsmodelle enthält, d. h. das Definitionen der Grundklassen und -kategorien enthält, die durch die eigene Taxonomie etabliert wurden. Das DTD- und XML-Schema des Data Consortiums z. B. enthält nur 160 Elemente. Diese beiden Schemata werden zur Validierung aller kodierten Dokumente verwendet, die den Namespace verwenden. (Ca. 160 Elemente stellen die 10 im Wörterbuch definierten Grundklassen dar, 140 Grundkategorien und 10 Datentypen).

Die zweite Stufe der Validierung vergleicht den Inhalt eines Dokuments mit Geschäftsregel-Metadaten, die im RDFS-Repository gespeichert sind. Die Geschäftsregeln werden mit Hilfe von RDF-Schema-Elementen, Agent Markup Language-Elementen (DAML+OIL) und Elementen ausgedrückt, die eine ECMA-Programmierumgebung für spezifische Dokumente vordefinieren. Das Inhaltsmodell für jeden Begriff wird sowohl vom Element „Property" des RDF-Schemas und unserem Element „ECMASlot" definiert. Es ist technisch auch machbar, ein dokumentenspezifisches DTD- oder XML-Schema direkt mit Informationen im RDF-Schema zu erstellen.

Ein wichtiger Vorteil dieses Ansatzes ist, dass aufgrund der Erwartung, dass Grundklassen und Kategorien relativ stabil sind, der Änderungsbedarf am DTD- und XML-Schema ebenfalls wahrscheinlich relativ gering sein wird. Weil XSL-T und CSS-Stylesheets mit Hilfe von mit diesen beiden Schemata definierten Elementen entwickelt werden (nicht durch ein RDF-Schema), benötigen diese Stylesheets umso weniger Pflege, je mehr das Geschäftsobjektmodell im Lauf der Zeit verfeinert wird. Software auf höherer Ebene wie ein generisches Dokumentenverwaltungssystem kann Elemente, die ein Dokument darstellen, direkt identifizieren, ohne den Satz der Dokumentnamen im Dokument mit aufzeichnen zu müssen.

Software, die eine detailliertere Auswahlfunktion erfordert, wird durch isA()-Verarbeitung kompatibel gemacht. Dabei handelt es sich um eine äußerst wichtige Funktion in einem Verarbeitungsmodell, das mehrere dynamische Eingaben wie bei RDF umfasst: der Typ eines gegebenen Elements wird durch (a) seinen Elementnamen, (b) seinen Attributwert *xsi:type*, (c) seinen Attributwert *name* und (d) seine Child-Elemente `rdf:type` angegeben [Hinweis: Der Toolkit des Data Consortiums fügt aus Leistungsgründen redundant Elemente `rdf:type` entsprechend (a) bis (c) ein, so dass eine einzige Zugriffsmethode im ganzen Toolkit verwendet wird.]

## *Damit zusammenhängende Aktivitäten*

Data Consortium-Toolkit (Open-Source)

- → Eingaben: (gepunkteter Name, dotted-name) HTML, SVG, XFORM und XScript-Datenströme
- → Eingaben: (Standard-XML) Data Consortium, Dublin Core u. IMC-Datenströme
- → führt aus: Client-eingereichte ECMA-Script-Programme
- → Ausgaben: semantische Validierungsmeldungen (Durchsetzung von Geschäftsregeln)
- → Ausgaben: DCN- und XScript-Datenströme

LEXML-Dictionary

- → Gemeinsame Bemühung mit US-basierter LegalXML-Dictionary-Arbeitsgruppe
- → Fokus auf mehrere nationale Jurisdiktionen

RDF-Schema-Wörterbücher

- → DARPA Dictionary for Web Service
- → Semantic Web Agreement Group (SWAG): allgemeiner Vertreiber

Das Data Consortium entwickelt jetzt einen CORBA-kompatiblen Toolkit, der zwei Arten von Eingabedatenströmen akzeptiert. Immer wenn ein Datenstrom gelesen wird, wird ein Objektmodell konstruiert, auf das entweder über CORBA-kompatible Application Program Interfaces (APIs) oder über eine Standard-ECMA-Umgebung zugegriffen wird. Diese Objektmodelle werden aus (a) Standard-XML-Elementen erzeugt, die vom Namespace von Data Consortium, von der Dublin Core-Organisation und vom Internet Mail Consortium (IMC) erstellt werden. Der Namespace von Dublin Coreumfasst 14 Elemente, die von Bibliothekswissenschaftlern entwickelt wurden und allgemein elektronische Ressourcen beschreiben. Der Namespace des IMC umfasst Electronic Business Card (vCard)-Elemente, die E-Mail-Empfänger beschreiben. In Zukunft kann der Toolkit mit weiteren Namespaces arbeiten. Der zweite vom Toolkit akzeptierte Eingabetyp sind Datenströme, die die Variation „gepunkteter Name“ für XML-Elementnamen verwenden, was vom Data Consortium als „XScript“ bezeichnet wird. HTML, SVG und XFORM-Datenströme, die selbst „gepunktete Namen“ verwenden, sind ebenfalls eine gültige Eingabe für den Toolkit.

RDF-Schema-basierte Wörterbücher werden für die juristische Gemeinde in den USA von LegalXM und für die in Europa von LexML entwickelt, unserer Schwesterorganisation, die von Murk Muller geleitet wird.

Falls Ihre Organisation an der aktuellen Entwicklung dieser Software beteiligt werden möchte, kontaktieren Sie bitte Murk Muller oder mich. Wir würden diese Produkte gerne so schnell wie möglich in die Public Domain bringen, deshalb wären wir Ihnen für die Mithilfe durch Ihre Fachleute sehr dankbar.