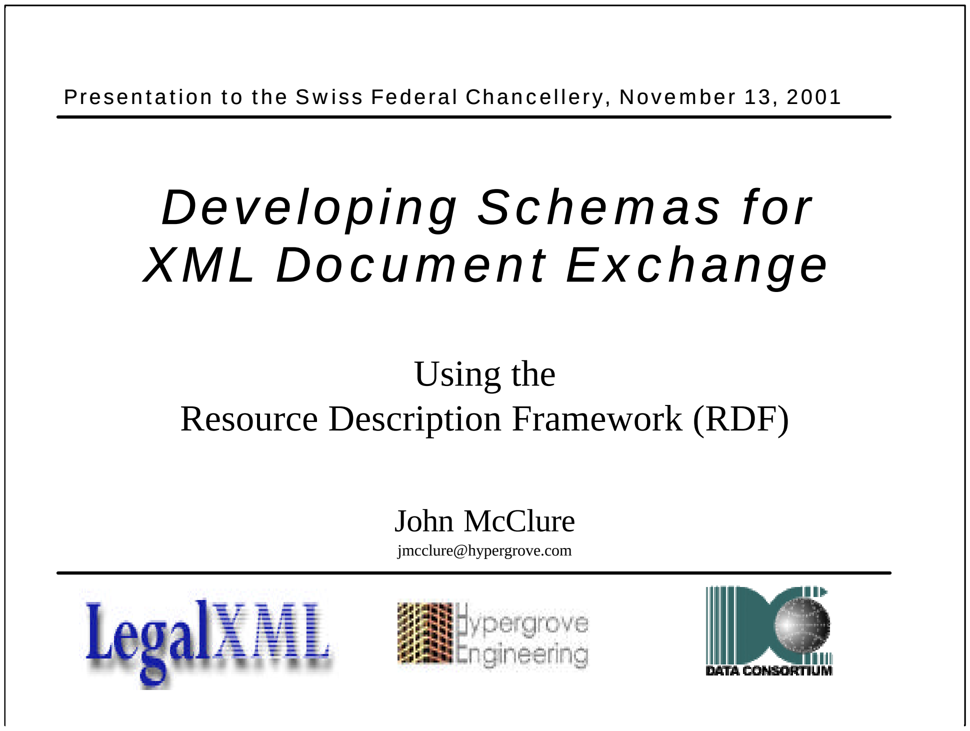

Presentation to the Swiss Federal Chancellery, November 13, 2001

Developing Schemas for XML Document Exchange

Using the
Resource Description Framework (RDF)

John McClure
jmcclure@hypergrove.com

Introduction. One consequence of using the Extensible Markup Language (XML) technology is a new focus on <u>text documents</u> exchanged between organizations and persons. Historically, software developers focused on binary messages exchanged between applications executing on client and server computers. Two key architectural questions occur: (1) what information within a document needs to be separately 'tagged' by XML elements, so it may be extracted and subsequently used; and (2) what are the optimal formats for, and arrangements of, XML elements within the document. The answer to the first question is represented by a **document schema**, indicating what is tagged within a document. The answer to the second question – how to code document instances using XML elements – is partially determined by the technology used to record a document's schema.

Two XML dialects exist for representing document schema : (1) *RDF Schema* and (2) *XML Schema*. One subject today is the <u>difference</u> between these dialects, both important Technical Recommendations from the Worldwide Web Consortium (W3C), meaning that they have the greatest official sanction possible from the W3C. The objective of this presentation is to help you to make an informed decision about the technologies most suitable to your organizations and software applications; my own aim is to share with you our standards' architecture, which <u>combines</u> the two schema dialects in a manner that effectively leverages their best properties.

Speaker. My name is John McClure, founder of Hypergrove Engineering, a firm whose goal is to create software used during the formation and management of lease and purchase contracts. My own objective has been to create open, public standards for legal information, ultimately becoming the architect for the Data Consortium, a non-profit association within the commercial real estate sector in the United States. I chair two workgroups within LegalXML, an association including courts, law enforcement, and legal publishers. I chair LegalXML's Contracts Workgroup and also, with Murk Muller, chair their Dictionary Workgroup, where we are creating a dictionary complying with the Resource Description Framework. My background is economics, which led me to the field of computer science. At IBM, I was a developer of the OS/2 operating system's desktop environment, and later the leader of a team that created an SGML document editor. At MCI, I was responsible for application and data architectures, a position in which I became an evangelist encouraging company-wide adoption of the Extensible Markup Language.

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

Metadata Specification

Characteristic	***RDF Schema***	***XML Schema***
Popular Name	“Dictionary”,“Ontology”	“Document Schema”
➡ General Orientation	Defines names for things Defines object models	Defines XML elements Defines a transport protocol

Basic Elements	`<Class>`
	`<Property>`
		`<Element>`
		`<Attribute>`

Basic Datatypes	XML Schema Datatypes	XML Schema Datatypes
Constraint Support	Yes	Yes
Inheritance Support	Multiple Inheritance	Single Inheritance

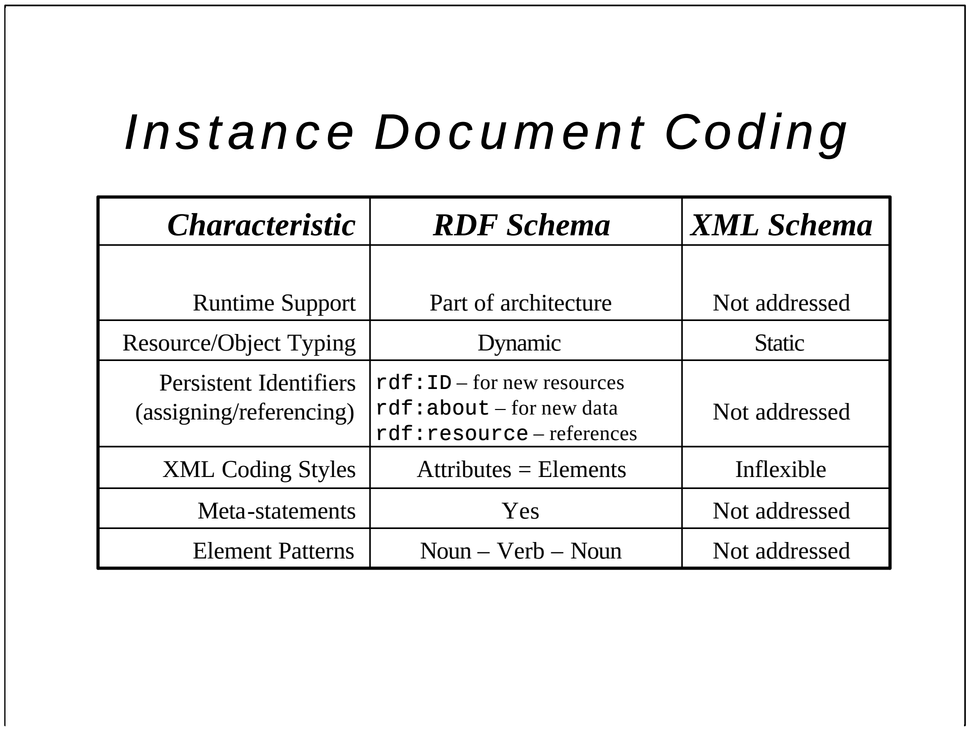
Synonym Support	`<sameClassAs>`
	`<samePropertyAs>`
		?

The objective of any schema is to define metadata in a manner useful to application software. Metadata includes documents that are within the scope of the software, fields on the documents, and permissible values for fields. RDF Schema (RDFS) and XML Schema (XMLS) are the XML dialects for metadata specification the W3C recommends, although others do exist (ASN, DDL, DTD, Relax NG, XMI). RDFS repositories are popularly called dictionaries or, more technically, semantic ontologies. XMLS grew from Document Type Definitions (DTDs), and its repositories are accordingly called “document schema’.

The key distinction between the two dialects is that **RDFS** defines names for **information elements** (using “Class” elements), while **XMLS** defines names for **XML elements** (uses “Element” elements). This is important. Because RDFS defines types of program objects manipulated by software, and XMLS defines a specific transport protocol for information read and written by software, RDFS dictionaries can be applied to multiple transport protocols; XMLS schemas cannot. An RDFS dictionary is applicable to an XML representation (acting like an XML Schema); an HTML/SVG/XFORMS representation; or even a binary representation. XML Schema cannot do this.

In order for RDFS metadata to be functionally equivalent to XMLS metadata, one needs to look deeper. Both provide familiar content models, constraint specifications; and the identical set of intrinsic data-types (strings, numerics, dates, URLs, etc.). Like most object-oriented programming languages, RDFS however does accommodate multiple inheritance, while XMLS does not. Further, RDFS does provide synonym and antonym support, concepts foreign to XMLS. RDFS is as capable as XMLS performing the basic functions of Document Type Definitions, the metadata facility they were each designed to replace – instance datastream validation.

Finally, given that an organization’s metadata repository is expected to be a large volume of information, the question must be asked: is it a good organizational investment to create and maintain two metadata repositories? The short answer is yes, if the size of the XMLS repository is consciously small, placing the majority of metadata within the RDFS repository. The longer answer involves recognizing the need to accommodate datastream transformation and query requirements.



Instance Document Coding

Characteristic	RDF Schema	XML Schema
Runtime Support	Part of architecture	Not addressed
Resource/Object Typing	Dynamic	Static
Persistent Identifiers (assigning/referencing)	`rdf:ID` – for new resources `rdf:about` – for new data `rdf:resource` – references	Not addressed
XML Coding Styles	Attributes = Elements	Inflexible
Meta-statements	Yes	Not addressed
Element Patterns	Noun – Verb – Noun	Not addressed

The Resource Description Framework provides more than just a metadata repository. Its architecture envisions a runtime environment that interprets RDF datastreams as a series of statements composed of a subject, a predicate, and an object (a tuple). RDF also accommodates meta-statements, that is, a statement whose object is another statement (example: "John said *to keep most metadata in an RDFS dictionary*"). The runtime facility therefore allows an application to "select" all subjects in a document, subject to the values of associated predicates and objects, similar to relational database selections.

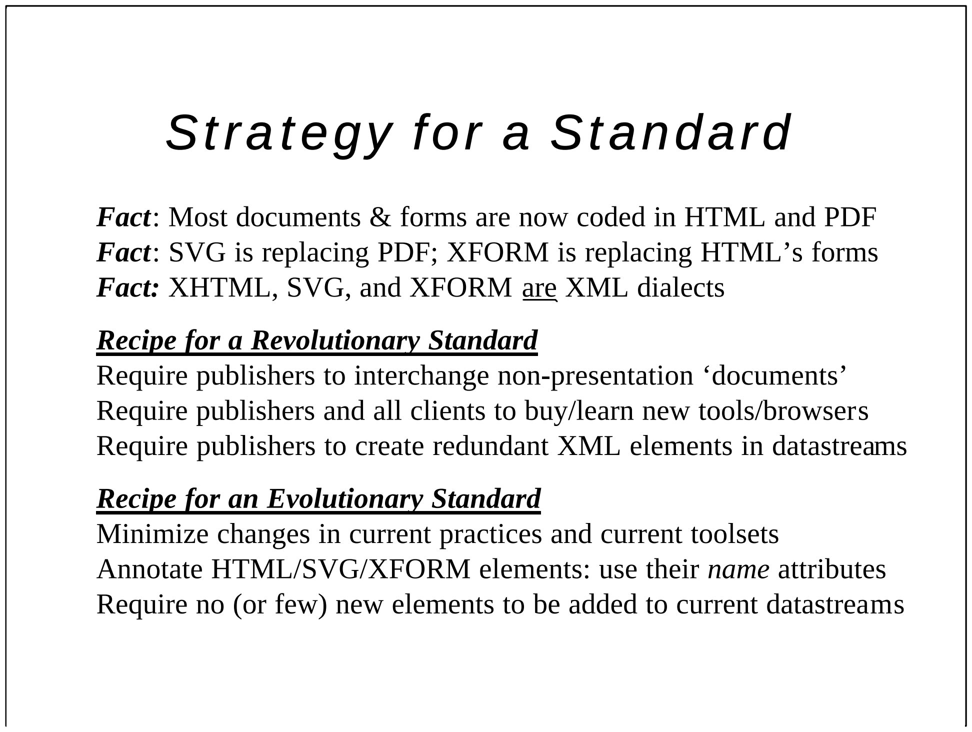
During database design and programming, problems occur when an entity has multiple types and the underlying technology allows only a single type to be associated with the entity. In the RDF, this problem is eliminated because multiple types can be associated with a single resource. The conventional approach for indicating that something is of multiple types is to use boolean elements (flags); in the RDF however the resource automatically assumes the attributes (or properties) associated with its types – thus elegantly matching the realities that are being modeled (see example below).

RDF also includes a mechanism for indicating when an XML element represents the definition of a new resource (use the `rdf:ID` attribute), or contains information that is about an existing resource (use the `rdf:about` attribute). RDF also addresses references of an existing resource (use `rdf:resource`).

RDF allows multiple encodings to have identical meanings. For instance, the `LawFirm` element below could be coded as `<LawFirm dc:Title=‘Smith &amp; Jones’/>.` The RDF runtime establishes the `dc:Title` information as an attribute (or property) of a `LawFirm` resource object, regardless of whether it was coded as an XML attribute or as an XML element.

Example RDF Mark-up

```
<Witness rdf:ID='http://www.uscourts.gov/USA.FederalCourt.CourtCase.Witness#AB1234567890'>
   <rdf:type rdf:resource=‘#Attorney’/>
   <represents rdf:resource=‘http://www.uscourts.gov/USA.FederalCourt.CourtCase.Defendant#FOO1234890’/>
   <employedBy>
     <LawFirm rdf:about=‘http://www.dot.net/USA.Business#AB1234567890’>
        <dc:Title>Smith &amp; Jones</dc:Title>
     </LawFirm>
   </employedBy>
</Witness>
```



Strategy for a Standard

Fact: Most documents & forms are now coded in HTML and PDF
Fact: SVG is replacing PDF; XFORM is replacing HTML's forms
Fact: XHTML, SVG, and XFORM <u>are</u> XML dialects

<u>Recipe for a Revolutionary Standard</u>
Require publishers to interchange non-presentation 'documents'
Require publishers and all clients to buy/learn new tools/browsers
Require publishers to create redundant XML elements in datastreams

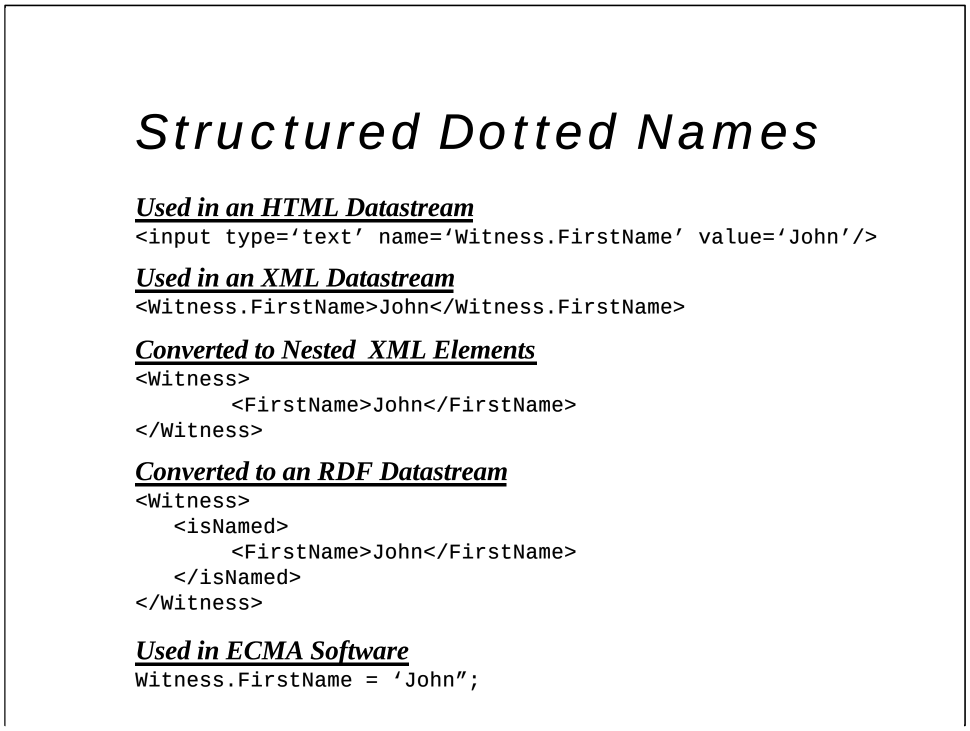
<u>Recipe for an Evolutionary Standard</u>
Minimize changes in current practices and current toolsets
Annotate HTML/SVG/XFORM elements: use their *name* attributes
Require no (or few) new elements to be added to current datastreams

The usual goal of vertical industry XML standard-making organizations is the exchange of <u>documents</u> that are created within the vertical industry. A "document" is <u>presentation material</u> that, until now, has been represented using HTML or Adobe's PDF language, both non-conforming with XML and both unquestionably in the process of being replaced by XML-compliant dialects: XHTML and XFORM are replacing HTML, and the Scalable Vector Graphics (SVG) language is replacing PDF. The implication is that all documents will soon be coded by publishers using XHTML, XFORM, and SVG.

Therefore, is it not ironic that standards-making consortia are creating non-presentation dialects of XML that are to be used to interchange presentation material? By definition, consortia seem to be creating standards for the exchange of <u>databases</u>, not documents, because the information is not in its final, substantial, form. The implication is that publishers committed to exchange need to create both presentation material <u>and</u> a database representing the document. That is unacceptable since the ultimate impact of the standard is to be maximally disruptive to current business practices which are (becoming) based on XHTML, SVG, and XFORM. The more disruption, the less likely the proposed standard will take root in stakeholder organizations.

The HTML, SVG, and XFORM dialects of XML must be leveraged as the foundation for document exchange. Fortunately, these three dialects already include the mechanics necessary for standards organizations to do this – most of their elements include a *name* attribute in which publishers can place a ***<u>structured dotted name</u>***. This dotted-name is used to label the information represented by the element.

In other words, standards consortia must focus on establishing the structure of dotted-names, not on establishing entirely new dialects of XML that are disjoint with respect to current business practices, tools, and skills, and are disjoint with respect to the objects they are intended to represent, that is, presentation documents.



Structured Dotted Names

Used in an HTML Datastream

```
<input type='text' name='Witness.FirstName' value='John'/>
```

Used in an XML Datastream

```
<Witness.FirstName>John</Witness.FirstName>
```

Converted to Nested XML Elements

```
<Witness>
       <FirstName>John</FirstName>
</Witness>
```

Converted to an RDF Datastream

```
<Witness>
   <isNamed>
       <FirstName>John</FirstName>
   </isNamed>
</Witness>
```

Used in ECMA Software

```
Witness.FirstName = 'John";
```

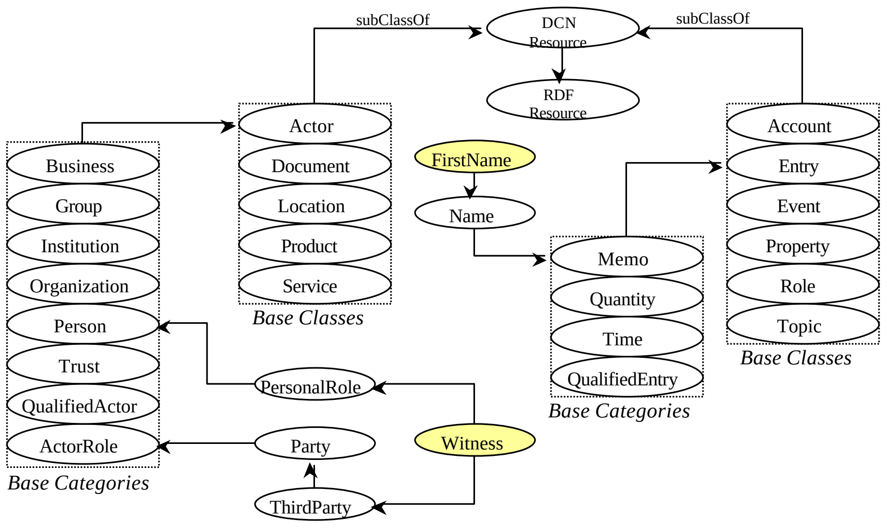
When used in an HTML datastream, a dotted-name is naturally used to label the value of the *value* attribute on an `<input>` element. Note that when the HTML form is posted to an HTTP server, the value of the *name* and *value* attributes are automatically extracted, giving the same result as if used in an XML datastream (the second scenario). However, there is a drawback from using a dotted name as the value of the *name* attribute in an HTML document. Dynamic HTML (Javascript) allows a seamless use of the *name* attribute as part of a property-name, meaning that Javascript programs would not be able to use this (somewhat odd) facility. If this is undesirable, then a namespace-qualified attribute could be used instead. For instance, using the Data Consortium's namespace, the attribute *dcn:name* would contain 'Witness.FirstName' while the *name* attribute would contain, for instance, "WitnessFirstName".

It is syntactically valid for the name of an XML element to be a dotted-name. However, note that when used as the name of an XML element, element nesting (see the third scenario) is eliminated – this has the beneficial effect of rendering the XML datastream itself easier to humanly comprehend and debug. In fact, little prevents adjacent XML elements which have no attribute values or text content from being allowed, by default, to be ***arbitrarily*** combined using the dot operator – the nested element structure can be easily inserted or removed mechanistically. Such pre-processing would be necessary in order to (a) validate the datastream using a DTD, an XML Schema, or an RDF Schema; and (b) process or otherwise transform the datastream using the Extensible Stylesheet Language (XSL-T).

In the next scenario, the dotted name is converted to an RDF representation. The predicate ("isNamed") is inserted between the nouns "Witness" and "FirstName" based upon metadata in the repository that identifies the relationships between nouns. The dotted name "Witness.isNamed.FirstName" is equally valid, however that name is potentially artificial and, if it were a publisher requirement, may encounter resistance from stakeholders wanting a simple non-intrusive standard for information exchange.

The same dotted name can be used in an ECMA Script program (Javascript). The "FirstName" property-slot of the "Witness" object is assigned a string value. Therefore, dotted names bring a consistency across several technologies now associated with the Internet. To achieve this end, XML Schema – used to define XML elements – is not appropriate. RDF Schema – used to define object classes and their properties – is well-suited for the repositories containing operational metadata.

Data Consortium Taxonomy

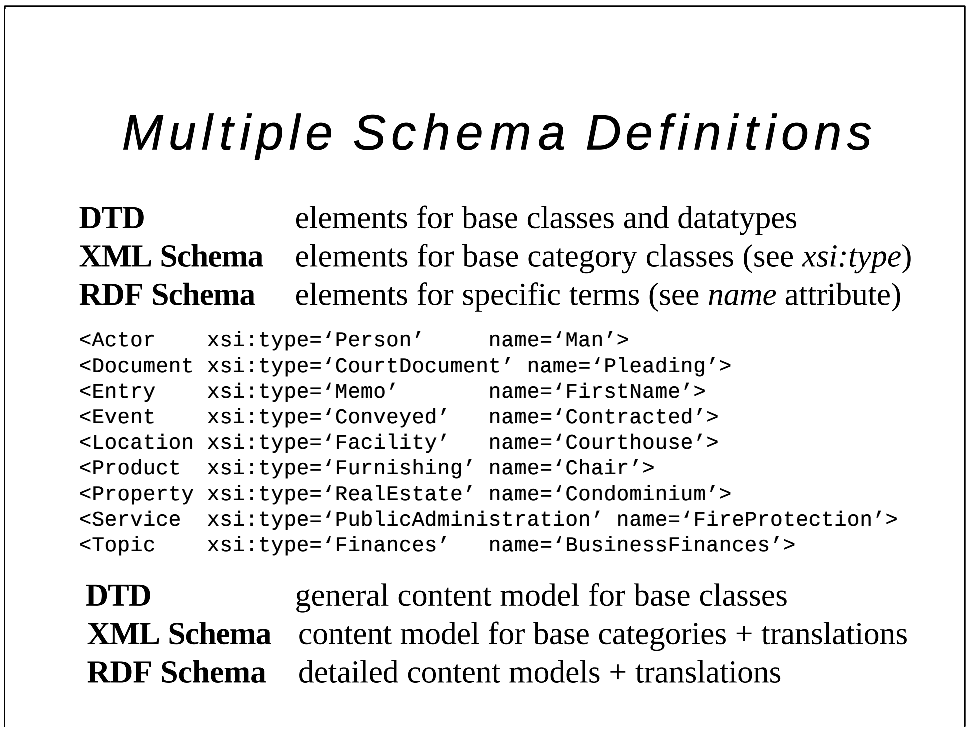


The Data Consortium's Dictionary contains a taxonomy for almost 8,000 terms related to the commercial real estate industry in the United States, a portion of which is displayed by this diagram. A *taxonomy* in this context is an arrangement of terms into super/subclass relationships. For instance, **DCNResource** is a subclass of RDF's **Resource** class, a class that represents a generic object in the Resource Description Framework. A superclass could be called a "category" however it is more than just a container for other terms – every object class has properties that can be assigned to it, and all properties assigned to a superclass are inherited by its subclasses. One can also say subclasses are sub-types (thus the `rdf:type` element is appropriately named). The arrows on the diagram point to a term's superclass.

In the Dictionary, a **FirstName** is a type of **Name**, which is a type of textual memorandum (**Memo**) that can be recorded about a resource. A **Memo** is a subtype of **Entry**, one of the eleven basic classes in the taxonomy. The **Memo** class is (called) a base category because it is an immediate subtype of a base class. Other base category classes within **Entry** include **Quantity** (for numeric data elements); **Time** (for time and date data elements); and **QualifiedEntry** (for names prefixed by a modifier term). The taxonomy for a **Witness** is less trivial: it is a subclass of both **ThirdParty** and **PersonalRole.** A **ThirdParty** is a subclass of **Party,** a subtype of **ActorRole** whose superclass is the **Actor** class**.** A **PersonalRole** is a subclass of a **Person,** also a subclass of the **Actor** class.

The objective of a taxonomy is to facilitate the definition and enforcement of business rules applicable to document content, minimizing programming required as a taxonomy is improved or corrected – this is crucial to standards development. For instance, because a **Witness** is a subtype of **Person,** then it may have a **FirstName** as any **Person** object can. Additionally, wherever a schema allows a **Witness** to be used, any of its subclasses may equally be used; for instance, a **HostileWitness** class could be inserted into the taxonomy as a subclass of **Witness,** and used without changing much if any existing software.

Modern programming systems require multiple class inheritance to correctly model the information that they manipulate. Because RDF Schema accommodates multiple inheritance, a facility not available with XML Schema, it is therefore a better foundation for a (distributed) metadata repository than XML Schema that includes both normative metadata plus organization-specific metadata.



Multiple Schema Definitions

DTD	elements for base classes and datatypes
XML Schema	elements for base category classes (see *xsi:type*)
RDF Schema	elements for specific terms (see *name* attribute)

```
<Actor    xsi:type='Person'     name='Man'>
<Document xsi:type='CourtDocument' name='Pleading'>
<Entry    xsi:type='Memo'       name='FirstName'>
<Event    xsi:type='Conveyed'   name='Contracted'>
<Location xsi:type='Facility'   name='Courthouse'>
<Product  xsi:type='Furnishing' name='Chair'>
<Property xsi:type='RealEstate' name='Condominium'>
<Service  xsi:type='PublicAdministration' name='FireProtection'>
<Topic    xsi:type='Finances'   name='BusinessFinances'>
```

DTD	general content model for base classes
XML Schema	content model for base categories + translations
RDF Schema	detailed content models + translations

The primary function of document schemas is to support validation of the content of XML documents. The ultimate aim of validation is to ensure that a document contains XML elements that are named in stylesheets and other software later processing the document. Our approach creates multiple stages for validation corresponding to the schema used. The first stage syntactically inspects the document to ensure it validates against a DTD or XML Schema containing definitions of broadly-defined data types and broadly-defined content models – in other words, that contains definitions of the base classes and base categories established by one's taxonomy. For instance, the Data Consortium's DTD and XML Schema contains only 160 elements, and these two schema are used to validate <u>all</u> documents encoded using the namespace. (Approximately 160 elements represent the 10 base classes defined in the Dictionary, 140 base categories, and 10 datatypes).

The second stage of validation compares the content of a document against business rule metadata stored in an RDFS repository. The business rules are expressed using RDF Schema elements, Agent Markup Language elements (DAML+OIL), and elements that pre-define an ECMA programming environment for specific documents. The content model for each term is defined by both RDF Schema's "Property" element and by our "ECMASlot" element. It is also technically feasible to generate a document-specific DTD or XML Schema directly from information in the RDF Schema.

A noteworthy advantage of this approach is that – since base classes and categories are anticipated to be relatively stable – the need to change the DTD and XML Schema is equally anticipated to be very small. Because XSL-T and CSS stylesheets are developed using elements defined by these two schema – not by an RDF Schema – then these stylesheets would need that much less maintenance as the business object model is refined over time. Higher-level software, such as a generic Document Management System, can directly identify elements that represent a document, without needing to have the set of names for documents be recorded within it.

Software that requires a more detailed selection capability is accommodated by isA() processing. This is a highly important facility in a processing model that includes multiple, dynamic, typing as provided by the RDF: a given element's type is indicated by (a) its element name (b) its *xsi:type* attribute value (c) its *name* attribute value and (d) its child `rdf:type` elements. [Note: the Data Consortium's toolkit, for performance reasons, redundantly inserts `rdf:type` elements corresponding to (a) through (c) so that a single access method is used throughout the toolkit.]

Related Activities

<u>Data Consortium Toolkit (Open-source)</u>

→ inputs: (dotted-name) HTML, SVG, XFORM, and XScript datastreams
→ inputs: (standard XML) Data Consortium, Dublin Core, and IMC datastreams
→ executes: client-submitted ECMA Script programs
→ outputs: semantic validation messages (business rule enforcement)
→ outputs: DCN datastreams and XScript datastreams

<u>LEXML Dictionary</u>

→ Fraternal effort with US-based LegalXML's Dictionary Workgroup
→ Focus on multiple national jurisdictions

<u>RDF Schema Dictionaries</u>

→ DARPA Dictionary for Web Service
→ Semantic Web Agreement Group (SWAG): general clearinghouse

The Data Consortium is now developing a CORBA-enabled toolkit which accepts two kinds of input datastreams. Whenever a datastream is read, an object model is constructed that can be accessed either by CORBA-enabled Application Program Interfaces (APIs) or through a standard ECMA environment. These object models are constructed from (a) standard XML elements that are defined by the Data Consortium's namespace, by the Dublin Core organization, and by the Internet Mail Consortium (IMC). The Dublin Core's namespace includes 14 elements that have been designed by library scientists which generically describe electronic resources. The IMC's namespace includes Electronic Business Card (vCard) elements that describe e-mail recipients. In the future, other namespaces can be accommodated by the Toolkit. The second type of input accepted by the Toolkit are datastreams that use the 'dotted-name' variation for XML element names, which is called "XScript" by the Data Consortium. And HTML, SVG, and XFORM datastreams using the 'dotted-name' themselves are also a valid input for the Toolkit.

RDF Schema-based dictionaries are being developed for the US legal community by LegalXML, and for the European legal community by LexML, our fraternal organization led by Murk Muller.

If your organization would like to be involved with the ongoing development of this software, please do contact me or Murk Muller. We would like to bring these products into the public domain as quickly as possible, so your expert participation (and funding!) would be greatly appreciated.