XML, RDF, DAML, and OWL

José M. Vidal

Tue Apr 20 13:19:52 EDT 2004

This talk summarizes material from the appropriate RFCs and W3C Recommendations, as well as:


• Deborah L. McGuinness and Frank van Harmelen editors, OWL Web Ontology Language Overview.

• Frank van Harmelen, Peter F. Patel-Schneider, and Ian Horrocks. DAML+OIL Walkthru.

1 XML

• eXtensible Markup Language. Defined in the W3C Recommendation.

• extensible means that it only provides a data format, not an actual vocabulary. XML is a metalanguage.

• Markup means that certain sequences of characters contain information indicating the role of the content.

• tags are those words between pointy brackets:

  <tag>

1.1 XML

• An element is the main XML entity: an opening and closing tag. They need to be properly nested.

  <name>Vidal</name>
  <person/>

  <b><i>bad nesting</i></b> <i></i>

• Elements can have attributes (id) which have values (123):
The values have to appear between quotes.

1.2 XML

• Comments are delimited between <!-- and -->

• **Processing instructions** allow a document to contain instructions to applications. They are of the form <? Instructions ?>.

• CDATA sections are used to escape characters that would otherwise be considered XML markup. They are of the form <![CDATA[ datahere ]]>.

• If you just want to escape a character it is better to use the character entities,

• All documents must begin with an XML declaration like <?xml version="1.0"?>

1.3 XML Example

• An example purchase order:

```xml
<?xml version="1.0"?>
<purchaseOrder orderDate="1999-10-20">
  <shipTo country="US">
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
    <city>Mill Valley</city>
    <state>CA</state>
    <zip>90952</zip>
  </shipTo>
  <billTo country="US">
    <name>Robert Smith</name>
    <street>8 Oak Avenue</street>
    <city>Old Town</city>
    <state>PA</state>
    <zip>95819</zip>
  </billTo>
  <comment>Hurry, my lawn is going wild!</comment>
  <items>
    <item partNum="872-AA">
      <productName>Lawnmower</productName>
      <quantity>1</quantity>
      <USPrice>148.95</USPrice>
      <comment>Confirm this is electric</comment>
    </item>
    <item partNum="926-AA">
      <productName>Baby Monitor</productName>
      <quantity>1</quantity>
      <USPrice>39.98</USPrice>
      <shipDate>1999-05-21</shipDate>
    </item>
  </items>
</purchaseOrder>
```
2 Document Type Definitions

- Document Type Definitions (DTDs) and XML Schemas specify the names of the elements and attributes and their use in documents.

- The DOCTYPE definition in a document specifies the one to be used, e.g. ```<!DOCTYPE greeting SYSTEM "hello.dtd">```.

- This sample DTD says that the element `questionlist` must have an attribute named `test` and contain only `question` elements.

```
<!ENTITY % Text "CDATA">

<!ELEMENT questionlist (question)+>
<!ATTLIST questionlist
  test %Text;>

<!ELEMENT question (qtext—choice—comment—%Text)+>
<!ELEMENT qtext (%Text)>
<!ELEMENT choice (%Text)>  
<!ATTLIST choice
  answer %Text; #IMPLIED>

<!ELEMENT comment (%body.content)>
```

3 XML Namespaces

- Defined in the [XML namespaces recommendation](#).

- An XML namespace is a collection of names, identified by a URI reference (RFC2396), which are used in XML documents as element types and attribute names.

- You use them when you need to use more than one DTD or Schema.

- A namespace is declared as using the `xmlns` attribute:

```
<x xmlns="http://www.w3.org/TR/REC-html40" xmlns:edi="http://ecommerce.org/schema">
  <!- the "edi" prefix is bound to http://ecommerce.org/schema for the "x" element and contents -->
  <edi:price units="Euro">32</edi:price>
  <lineItem edi:taxClass="exempt">Baby food</lineItem>
</x>
```

- The `edi` is called the namespace prefix.

- After this declaration and within the `x` element any tag `tagname` that is appears as `edi:tagname` will be considered part of the `edi` schema.

- If the prefix is missing then that namespace becomes the default namespace.
Namespace scopes are nested:

```xml
<?xml version="1.0"?>
<!-- initially, the default namespace is "books" -->
<book xmlns="urn:loc.gov:books"
     xmlns:isbn="urn:ISBN:0-395-36341-6">
  <title>Cheaper by the Dozen</title>
  <isbn:isbn>1568491379</isbn:isbn>
  <notes>
    <!-- make HTML the default namespace for some commentary -->
    <p xmlns="urn:w3-org-ns:HTML">
      This is a <i>funny</i> book!
    </p>
  </notes>
</book>
```

4 XML Schema

- **XML Schema** is the successor to DTDs.
- It provides a richer grammar for prescribing the structure of elements.
  - For example, you can specify the exact number of allowed occurrences of child elements, or specify default values.
- They provide data typing.
- They provide inclusion and derivation mechanisms. Reuse common elements.
- They are written in XML. The structures of XML Schema are themselves described using an XML Schema.
- The following is the schema definition for our PurchaseOrder

```xml
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Purchase order schema for Example.com.
      Copyright 2000 Example.com. All rights reserved.
    </xsd:documentation>
  </xsd:annotation>

  <xsd:element name="purchaseOrder" type="PurchaseOrderType"/>
  <xsd:element name="comment" type="xsd:string"/>

  <xsd:complexType name="PurchaseOrderType">
    <xsd:sequence>
      <xsd:element name="shipTo" type="USAddress" />
      <xsd:element name="billTo" type="USAddress" />
      <xsd:element ref="comment" minOccurs="0"/>
      <xsd:element name="items" type="Items"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```
<xsd:complexType name="USAddress">
    <xsd:sequence>
        <xsd:element name="name" type="xsd:string"/>
        <xsd:element name="street" type="xsd:string"/>
        <xsd:element name="city" type="xsd:string"/>
        <xsd:element name="state" type="xsd:string"/>
        <xsd:element name="zip" type="xsd:decimal"/>
    </xsd:sequence>
    <xsd:attribute name="country" type="xsd:NMTOKEN" fixed="US"/>
</xsd:complexType>

<xsd:complexType name="Items">
    <xsd:sequence>
        <xsd:element name="item" minOccurs="0" maxOccurs="unbounded">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="productName" type="xsd:string"/>
                    <xsd:element name="quantity">
                        <xsd:simpleType>
                            <xsd:restriction base="xsd:positiveInteger">
                                <xsd:maxExclusive value="100"/>
                            </xsd:restriction>
                        </xsd:simpleType>
                    </xsd:element>
                    <xsd:element name="USPrice" type="xsd:decimal"/>
                </xsd:sequence>
                <xsd:attribute name="partNum" type="SKU" use="required"/>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>

<!-- Stock Keeping Unit, a code for identifying products -->
<xsd:simpleType name="SKU">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="^[d]{3}[-][A-Z]{2}$"/>
    </xsd:restriction>
</xsd:simpleType>
4.1 XML Schema Types

- A primitive data type cannot be expressed in terms of any other data type.
- A derived type might be composed of other types.
- Data types are derived by restriction or extension of existing data types.

4.2 Building XML Schema Types

- It restricts string. -->

```xml
<?xml version="1.0"?>

<!- A product code is 2 digits, a dash, and five digits-->
<!- It restricts string. -->
<xsd:simpleType xmlns:xsd="http://www.w3.org/2000/10/XMLSchema" name="productCode">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d\{2\}-\d\{5\}"/>
  </xsd:restriction>
</xsd:simpleType>

<!- We can extend this product code by allowing an optional -->
<!- dash followed by a letter -->
<xsd:simpleType name="productCodeEx">
  <xsd:restriction base="productCode">
    <xsd:pattern value="\d\{2\}-\d\{5\}[-]*[a-z]\{0,1\}"/>
  </xsd:restriction>
</xsd:simpleType>
```
4.4 XML Complex Types

- We can define a schema for phone numbers that are just three group of numbers.

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- A telephone number contains three parts-->
<!-- This is the schema -->
<xsd:complexType name="telephoneNumber">
  <xsd:sequence>
    <xsd:element name="area">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:pattern value="\d{3}"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="exchange">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:pattern value="\d{3}"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="number">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:pattern value="\d{4}"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

<!-- An instance of this phonenumber would look like-->
<telephone xsi:type="abc:telephoneNumber">
  <area>123</area>
  <exchange>123</exchange>
  <number>1234</number>
</telephone>
```

4.5 XML Complex Types by Extension

- If we wanted to add a country code to the previous telephoneNumber type we would extend it.

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- The new type-->
<xsd:complexType name="telephoneNumberEx">
  <xsd:complexContent>
    <xsd:extension base="telephoneNumber">
      <xsd:sequence base="telephoneNumber">
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```
<xsd:element name="countryCode">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[d]{2}"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
</xsd:sequence>
</xsd:extension>
</xsd:complexType>

<!--An instance of the type--> 
<telephone xs:s:type="abc:telephoneNumber">
  <area>123</area>
  <exchange>123</exchange>
  <number>1234</number>
  <countryCode>01</countryCode>
</telephone>

- Notice that the countryCode has to appear as the last element, because we are deriving by extension.
- The types declared in the base class must appear before those in the derived classes.

5 RDF

- Resource Description Framework.
- A model for representing data about resources on the web.
- Model is based on Subject, Verb, Object triples.
- A statement is an instantiated RDF triple.
- The Subject is a resource or statement, the Object can either be a resource or a literal (string). This allows us to express opinions about opinions.

5.1 RDF Picture

- RDF statements can be viewed as nodes and arcs diagrams.
- The nodes (ovals) represent resources. The arcs represent named properties. String literals are represented by rectangles.
- Ora Lassila is the creator of the resource http://www.w3.org/Home/Lassila.

- In RDF text this is represented by:
5.2 RDF Abbreviated Syntax

- This same example:

```xml
<?xml version="1.0"?>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:s="http://description.org/schema/"
     Description about="http://www.w3.org/Home/Lassila">
  <s:Creator>Ora Lassila</s:Creator>
</Description>
</RDF>
```

- Can be represented using a more abbreviated syntax as:

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://description.org/schema/"
         rdf:Description about="http://www.w3.org/Home/Lassila"
         s:Creator="Ora Lassila" />
</rdf:RDF>
```

- Similarly,

```xml
<!-- These statements-->
<rdf:RDF>
  <rdf:Description about="http://www.w3.org">
    <s:Publisher>World Wide Web Consortium</s:Publisher>
    <s:Title>W3C Home Page</s:Title>
    <s:Date>1998-10-03T02:27</s:Date>
  </rdf:Description>
</rdf:RDF>
```

- are equivalent to these ones-->

```xml
<rdf:RDF>
  <rdf:Description about="http://www.w3.org"
                  s:Publisher="World Wide Web Consortium"
                  s:Title="W3C Home Page"
                  s:Date="1998-10-03T02:27" />
</rdf:RDF>
```
5.3 RDF Example

- The individual referred to by employee id 85740 is named Ora Lassila and has the email address lassila@w3.org. The resource http://www.w3.org/Home/Lassila was created by this individual.

- Is represented by,

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator rdf:resource="http://www.w3.org/staffId/85740"/>
  </rdf:Description>

  <rdf:Description about="http://www.w3.org/staffId/85740">
    <v:Name>Ora Lassila</v:Name>
    <v:Email>lassila@w3.org</v:Email>
  </rdf:Description>
</rdf:RDF>
```

5.4 RDF Schema

- RDF Schema is a type system for RDF. It provides a way to define domain-specific properties and classes of resources.

- The basic modeling primitives are class, property and ConstraintProperty. They are all Resources.

- RDF Schemas, when used, provide a standard (if limited) model for describing facts about web resources.
5.5 RDF Schema Class Hierarchy

- Notice that there are subclass-of and type-of relationships.

5.6 RDF Schema Instance Example

- Is represented by the following RDF instance,

```xml
<rdf:RDF xml:lang="en"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <!-- Note: this RDF schema would typically be used in RDF instance data by referencing it with an XML namespace declaration, for example xmlns:xyz="http://www.w3.org/2000/03/example/vehicles#". This allows us to use abbreviations such as xyz:MotorVehicle to refer unambiguously to the RDF class 'MotorVehicle'. -->

  <rdf:Description ID="MotorVehicle">
    <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
  </rdf:Description>

  <rdf:Description ID="PassengerVehicle">
    <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
  </rdf:Description>

  <rdf:Description ID="Truck">
    <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  </rdf:Description>
</rdf:RDF>
```
<rdf:RDF>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description ID="Van">
  <rdfs:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description ID="MiniVan">
  <rdfs:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#Van"/>
  <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
</rdf:Description>

</rdf:RDF>

6 OWL

- OWL is the Web Ontology Language.

  "An ontology is the attempt to formulate an exhaustive and rigorous conceptual
  schema within a given domain, a typically hierarchical data structure containing all
  the relevant entities and their relationships and rules (theorems, regulations) within
  that domain" – Wikipedia:Ontology

- It is a W3C Recommendation.
- Also, see the OWL Web Ontology Language Overview
- The standard for writing semantics.

6.1 DAML+OIL

- The pre-cursor to OWL. OWL copies it almost to the letter.
- Darpa Agent Markup Language along with the Ontology Interexchange Language extend
  RDF in order to create a language for describing ontologies.
- There exist many ontologies written in DAML+OIL available for your use.
- You should always first try to reuse or extend existing ontologies.
- There is a W3C Note on DAML+OIL which provides a good overview.

6.2 OWL Classes

- Start by defining classes which are subsets that contain all objects of that type.
This is the definition of an Animal class, with a label of Animal and two subclasses: Male and Female.

6.3 OWL Properties

- An OWL property connects two items.

- They are datatype properties (owl:DatatypeProperty) if they are relations between instances of classes and RDF literals and XML Schema datatypes.

- object properties (owl:ObjectProperty) if they are relations between instances of two classes.

An Animal can have a parent which must be an Animal.

hasFather is a property that is a kind of hasParent property, i.e., x’s father is also x’s parent.
6.4 OWL Property Restrictions

- We define a restricted class Person

```xml
<owl:Class rdf:ID="Person">
  <rdfs:subClassOf rdf:resource="#Animal"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasParent"/>
      <owl:toClass rdf:resource="#Person"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

- A Person is an Animal. The parent of a Person is also a Person.

```xml
<rdfs:subClassOf>
  <owl:Restriction owl:cardinality="1">
    <owl:onProperty rdf:resource="#hasFather"/>
  </owl:Restriction>
</rdfs:subClassOf>

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

- Any person must have exactly 1 father and at least one shoe size.

6.5 OWL Extends Other Ontologies

- You can extend an existing ontology by simply saying things about the terms in it.

```xml
<owl:Class rdf:about="#Animal">
  <rdfs:comment>
    Animals have exactly two parents, ie:
    If x is an animal, then it has exactly 2 parents
    (but it is NOT the case that anything that has 2
    parents is an animal).
  </rdfs:comment>
  <rdfs:subClassOf>
    <owl:Restriction owl:cardinality="2">
      <owl:onProperty rdf:resource="#hasParent"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

- If the ontology is already published, you would instead use the full URL.
6.6 OWL Defines Individuals

- You can also define individual objects in a class.

```xml
<Person rdf:ID="Adam">
    <rdfs:label>Adam</rdfs:label>
    <rdfs:comment>Adam is a person.</rdfs:comment>
    <age><xsd:integer rdf:value="13"/></age>
    <shoesize><xsd:decimal rdf:value="9.5"/></shoesize>
</Person>

- Adam is a Person.
### 6.7 OWL Ontology

<table>
<thead>
<tr>
<th>RDF Schema Features:</th>
<th>(In)Equality:</th>
<th>Property Characteristics:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class (Thing, Nothing)</td>
<td>equivalentClass</td>
<td>ObjectProperty</td>
</tr>
<tr>
<td>rdfs:subClassOf</td>
<td>equivalentProperty</td>
<td>DatatypeProperty</td>
</tr>
<tr>
<td>rdf:Property</td>
<td>sameAs</td>
<td>inverseOf</td>
</tr>
<tr>
<td>rdfs:subPropertyOf</td>
<td>differentFrom</td>
<td>TransitiveProperty</td>
</tr>
<tr>
<td>rdfs:domain</td>
<td>AllDifferent</td>
<td>SymmetricProperty</td>
</tr>
<tr>
<td>rdfs:range</td>
<td>distinctMembers</td>
<td>FunctionalProperty</td>
</tr>
<tr>
<td>Individual</td>
<td></td>
<td>InverseFunctionalProperty</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property Restrictions:</th>
<th>Restricted Cardinality:</th>
<th>Header Information:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction</td>
<td>minCardinality (only 0 or 1)</td>
<td>Ontology</td>
</tr>
<tr>
<td>onProperty</td>
<td>maxCardinality (only 0 or 1)</td>
<td>imports</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>cardinality (only 0 or 1)</td>
<td></td>
</tr>
<tr>
<td>someValuesFrom</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class Intersection:</th>
<th>Versioning:</th>
<th>Annotation Properties:</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>versionInfo</td>
<td>rdfs:label</td>
</tr>
<tr>
<td></td>
<td>priorVersion</td>
<td>rdfs:comment</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rdfs:seeAlso</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rdfs:isDefinedBy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AnnotationProperty</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OntologyProperty</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Datatypes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>xsd datatypes</td>
</tr>
</tbody>
</table>

### 6.8 OWL Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
  xmlns:rdfs="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:camera="http://www.xfront.com/owl/ontologies/camera/#">
  <xml:base href="http://www.xfront.com/owl/ontologies/camera/>
</rdf:RDF>
```
<owl:Ontology rdf:about="">
  <rdfs:comment>
Camera OWL Ontology
Author: Roger L. Costello
</rdfs:comment>
</owl:Ontology>

<owl:Class rdf:ID="Money">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>

<owl:DatatypeProperty rdf:ID="currency">
  <rdfs:domain rdf:resource="#Money"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="Range">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>

<owl:DatatypeProperty rdf:ID="min">
  <rdfs:domain rdf:resource="#Range"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="max">
  <rdfs:domain rdf:resource="#Range"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="units">
  <rdfs:domain rdf:resource="#Range"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="Window">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>

<owl:Class rdf:ID="Viewer">
  <owl:oneOf rdf:parseType="Collection">
    <camera:Window rdf:about="#ThroughTheLens"/>
    <camera:Window rdf:about="#WindowOnTopOfCamera"/>
  </owl:oneOf>
</owl:Class>

<owl:Class rdf:ID="PurchaseableItem">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="cost"/>
<rdfs:domain rdf:resource="#PurchaseableItem"/>
<rdfs:range rdf:resource="#Money"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="Body">
  <rdfs:subClassOf rdf:resource="#PurchaseableItem"/>
</owl:Class>

<owl:Class rdf:ID="BodyWithNonAdjustableShutterSpeed">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class about="#Body"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#shutter-speed"/>
      <owl:cardinality>0</owl:cardinality>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>

<owl:Class rdf:ID="Lens">
  <rdfs:subClassOf rdf:resource="#PurchaseableItem"/>
</owl:Class>

<owl:Class rdf:ID="Camera">
  <rdfs:subClassOf rdf:resource="#PurchaseableItem"/>
</owl:Class>

<owl:Class rdf:ID="SLR">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class about="#Camera"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#viewFinder"/>
      <owl:hasValue rdf:resource="#ThroughTheLens"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>

<owl:Class rdf:ID="Large-Format">
  <rdfs:subClassOf rdf:resource="#Camera"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#body"/>
      <owl:allValuesFrom rdf:resource="#BodyWithNonAdjustableShutterSpeed"/>
    </owl:Restriction>
  </owl:subClassOf>
</owl:Class>

<owl:Class rdf:ID="Digital">
  <rdfs:subClassOf rdf:resource="#Camera"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="part"/>

<owl:ObjectProperty rdf:ID="lens"/>
  <rdfs:subPropertyOf rdf:resource="#part"/>
6.9 From DAML+OIL to OWL

- The changes are listed in Appendix D of the OWL reference\textsuperscript{15}. They are minor.
- In fact, a student has written a [DAML+OIL to OWL converter]\textsuperscript{16}. 

15 http://www.w3.org/2003/07/ontology/ontol-0103246949

16 http://www.w3.org/2003/07/ontology/ontol-0103246949
Some of them are:

1. The namespace was changed to http://www.w3.org/2002/07/owl\textsuperscript{17}.

2. Various updates to RDF and RDF Schema from the RDF Core Working Group\textsuperscript{18} were incorporated, including:
   - cyclic subclasses are now allowed
   - multiple rdfs:domain and rdfs:range properties are handled as intersection
   - RDF Semantics
   - datatypes
     - RDF and OWL use the XML Schema namespace http://www.w3.org/2001/XMLSchema\textsuperscript{19} rather than http://www.w3.org/2000/10/XMLSchema\textsuperscript{20}.
     - OWL does not support using datatypes as types, e.g.

   ```xml
   <size>
     <xsd:integer rdf:value="10"/>
   </size>
   ```

   Instead use

   ```xml
   <size rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">10</size>
   ```

   - the daml:List construct used to represent closed collections was largely incorporated into RDF
     - rdf:parseType="Collection" replaces rdf:parseType="daml:collection"
     - daml:item is not supported.

3. Qualified restrictions were removed from the language, resulting in the removal of the following properties:
   - daml:cardinalityQ
   - daml:hasClassQ
   - daml:maxCardinalityQ
   - daml:minCardinalityQ

4. Various properties and classes were renamed, as shown in the following table:

<table>
<thead>
<tr>
<th>DAML+OIL</th>
<th>OWL</th>
</tr>
</thead>
<tbody>
<tr>
<td>daml:differentIndividualFrom</td>
<td>owl:differentFrom</td>
</tr>
<tr>
<td>daml:equivalentTo</td>
<td>owl:sameAs</td>
</tr>
<tr>
<td>daml:sameClassAs</td>
<td>owl:equivalentClass</td>
</tr>
<tr>
<td>daml:samePropertyAs</td>
<td>owl:equivalentProperty</td>
</tr>
<tr>
<td>daml:hasClass</td>
<td>owl:someValuesFrom</td>
</tr>
<tr>
<td>daml:toClass</td>
<td>owl:allValuesFrom</td>
</tr>
<tr>
<td>daml:UnambiguousProperty</td>
<td>owl:InverseFunctionalProperty</td>
</tr>
<tr>
<td>daml:UniqueProperty</td>
<td>owl:FunctionalProperty</td>
</tr>
</tbody>
</table>
5. `owl:SymmetricProperty` was added.

6. An `owl:DatatypeProperty` may be an `InverseFunctionalProperty` in OWL Full.

7. Synonyms for RDF and RDF Schema classes and properties were removed from the language, resulting in the removal of:
   - `daml:comment`
   - `daml:domain`
   - `daml:label`
   - `daml:isDefinedBy`
   - `daml:Literal`
   - `daml:Property`
   - `daml:range`
   - `daml:seeAlso`
   - `daml:subClassOf`
   - `daml:subPropertyOf`
   - `daml:type`
   - `daml:value`

8. `daml:disjointUnionOf` was removed from the language, since it can be effected using `owl:unionOf` or `rdfs:subClassOf` and `owl:disjointWith`.

9. `daml:equivalentTo` was renamed to `owl:sameAs`, and is no longer a superproperty of `owl:equivalentClass` and `owl:equivalentProperty`.

10. The following properties and classes were added to support versioning:
   - `owl:backwardCompatibleWith`
   - `owl:DeprecatedClass`
   - `owl:DeprecatedProperty`
   - `owl:incompatibleWith`
   - `owl:priorVersion`

11. `owl:AllDifferent` and `owl:distinctMembers` were added to address the Unique Names Assumption.

6.10 Three Sublanguages

- **OWL Lite** supports those users primarily needing a classification hierarchy and simple constraints.
- **OWL DL** supports those users who want the maximum expressiveness while retaining computational completeness (we use [Jess](https://www.jessrules.com)).
  - Named for Description Logics.
  - Supports all constructs, but sometimes only under certain conditions.
- **OWL Full** is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees.
  - It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.
6.11 The Future?

- OWL has world-wide support.
- We need services an applications that use OWL.
- Tools exist that do inferencing over OWL, but inferencing is boring. Induction anyone?.
- The technology is still in its infancy.

7 Summary

- XML provides the syntax.
- RDF provides semantically simple but common (as in "common-denominator") Subject Verb Action triples.
- OWL provides the power needed to describe useful ontologies.
- Each one extends the previous one.

Notes

http://jmvidal.cse.sc.edu/library/x2026.pdf
http://jmvidal.cse.sc.edu/library/w5063.pdf
http://www.w3.org/TR/owl-features/
http://www.daml.org/2001/03/daml+oil-walkthru.html
http://www.w3.org/TR/REC-xml
http://www.w3.org/TR/REC-xml-names/
http://www.ietf.org/rfc/rfc2396.txt
http://www.w3.org/XML/Schema
http://www.w3.org/TR/RDF/
http://www.w3.org/TR/rdf-schema/
http://www.wikipedia.org/wiki/Ontology_(computer_science)
http://www.w3.org/TR/owl-features/
http://www.daml.org/ontologies/
http://www.daml.org/ontologies/
http://www.daml.org/2002/06/webont/owl-ref-proposed#appd
http://www.w3.org/2002/07/owl
http://www.w3.org/2001/sw/RDFCore/
http://www.w3.org/2001/XMLSchema
http://www.w3.org/2000/10/XMLSchema
http://herzberg.ca.sandia.gov/jess/
http://www.mindswap.org/2000/10/XMLSchema

This talk is available at http://jmvidal.cse.sc.edu/talks/xmlrdfdaml

Copyright © 2004 Jose M Vidal. All rights reserved.