Combining RDF and XML Schemas to Enhance Interoperability Between Metadata Application Profiles

Jane Hunter
DSTC Pty Ltd
University of Qld, Australia
jane@dstc.edu.au

Carl Lagoze
Digital Library Research Group
Cornell University, NY
lagoze@cs.cornell.edu

Abstract

The term “application profile” has recently become highly topical. Heery and Patel [1] define application profiles as metadata schemas which consist of metadata elements drawn from one or more namespaces, combined together by implementers and optimised for a particular local application. They state that the principal characteristics of an application profile are that: it may draw on one or more existing namespaces; does not introduce new metadata elements; it can specify permitted schemes and values; and it can refine standard metadata elements. Significant new initiatives such as TV-Anytime [2], MPEG-21 [3] and the Open Archives Initiative (OAI) [4] are demanding application profiles which combine elements from a number of different existing standardized metadata schemas whilst maintaining interoperability and satisfying their own specific requirements through refinements, extensions and additions.

So far approaches to application profiles have been based on either RDF Schemas [5] or XML Schemas [6,7,8]. The SCHEMAS project [9] has adopted a purely RDF Schema approach. Justification for a pure XML Schema approach to application profiles is given in [10]. Despite high level assurances of unification from the W3C [11, 12], a purist and competitive attitude has prevailed amongst implementers. This has been because the demarcation of roles and the interface between these two disparate W3C Candidate Recommendations has been fuzzy; no low level details or implementations describing interface mechanisms have been provided; and implementers have been afraid of compromising interoperability. In this paper we describe a hybrid collaborative approach which combines the semantic knowledge of RDF Schemas with the explicit structural, cardinality and datatyping constraints provided by XML Schemas in a complementary manner. First we describe our view of how XML Schema and RDF Schema fit into the overall web metadata architecture. We then describe possible schema interface mechanisms. Finally using examples and mapping implementations based on XSLT and a metadata ontology, we demonstrate how interoperability between application profiles can be enhanced by using a dual schema approach.

Keywords: Metadata, Interoperability, XML, RDF, Schema, XSLT

1. Introduction

Metadata interoperability is a fundamental requirement for access to information on the Internet. In particular there are three scenarios in which interoperability between metadata descriptions is essential:

- To apply a single query syntax over descriptions expressed in multiple descriptive formats;
- To express the relationship between multiple descriptions in terms of a “core” or “canonical” description;
- To project community or individual specific descriptions out of a single canonical description.

The metadata interoperability problem has been exacerbated by the need for more complex metadata descriptions. It has become increasingly evident that simple standards such as Dublin Core (DC) [13] cannot satisfy the requirements of communities such as TV-Anytime [2], MPEG-21 [3], BIBLINK [14] and OAI [4] who need to combine metadata standards for simple resource discovery (DC), rights management (INDECS [15]), multimedia (MPEG-7 [16]), geospatial (FGDC [17]), educational (GEM [18], IEEE LOM [19]) and museum (CIDOC CRM [20]) content, to satisfy their application-specific requirements.

In this paper we propose mechanisms for metadata interoperability based on both RDF Schema and XML Schema. Using examples and implementations, we demonstrate how these two schema languages can be made to work together to enable flexible, dynamic mapping between complex, metadata descriptions which mix elements from multiple domains, i.e., *application profiles*. Our objective is to demonstrate how these two W3C Candidate Recommendations can be used in a complementary manner, exploiting the benefits of both.

In Section 2 we describe our overall web metadata architecture proposal and how the various components described in this paper fit together. In Section 3 we describe alternative mechanisms by which the two schema languages can be made to work together. The first part of section 3 defines clear boundaries between the responsibilities of RDF Schema and XML Schema to prevent functional overlap which could lead to contradictory constraints or incompatibilities. The second part of section 3 examines alternative mechanisms for linking complementary RDF and XML Schemas which are being used together to define a single metadata element set. In Section 4 we describe MetaNet, a “super-ontology” derived by merging a number of different domain-specific RDF Schemas. In Section 5 we describe how the semantic knowledge within MetaNet can be linked to XSLT to enable interoperability between application-profiles. Section 6 concludes with an overview of the advantages and disadvantages of this approach and the areas which require further work.

2. Semantic Web Metadata Architecture

In this section we propose a Web metadata architecture which will enable interoperability between domain-specific metadata schemas and application profiles consisting of metadata elements drawn from those schemas.

We propose that both metadata diversity and interoperability can more easily be accommodated across the WWW if each metadata domain defines both an RDF Schema and an XML Schema for their domain in their registered namespace. The RDF Schema file will define the domain-specific semantic knowledge by specifying type hierarchies and definitions - based on the ISO/IEC 11179 standard for the description of data elements. The XML Schema file will specify recommended encodings of metadata elements and descriptions by defining types and elements, and their content models, structures, occurrence constraints and datatypes. In addition, the XML Schema will contain links to the corresponding semantic definitions in the RDF Schema file in the same namespace. By expressing the semantic knowledge of each domain in a machine-understandable RDF Schema, it then becomes possible to merge these separate domain ontologies or vocabularies into a single encompassing ontology or vocabulary, also expressed as an RDF Schema, known as the MetaNet ontology.
XSLT provides the language for transforming between XML-encoded metadata descriptions. Combined with the semantic knowledge provided by MetaNet, XSLT is capable of performing both the semantic mapping and the structural and syntactic mapping required between metadata descriptions based on mixed-domain application profiles.
Hence the key components of this architecture, as illustrated in Figure 1, are:

- Domain-specific namespaces which express each domain's metadata model and vocabulary using both an RDF Schema and an XML Schema. Each XML Schema contains links to the corresponding RDF Schema;
- MetaNet - a single metadata ontology, expressed as an RDF Schema and based on a common underlying, extensible vocabulary. This has been generated by merging the domain-specific ontologies (RDF Schemas) from each namespace;
- XSLT - a language for transforming between XML-encoded metadata descriptions. When combined with the semantic knowledge of MetaNet, XSLT is capable of flexible dynamic mappings between application profile instantiations;
- Application Profiles - XML Schema definitions which combine, restrict, extend and redefine elements from multiple existing namespaces. In addition, using mechanisms such as those described in the next section, application profiles can also embed RDF Schema definitions of new Classes or Properties which are subClasses or subProperties of classes and properties defined in the domain-specific RDF Schemas.

In the next section we describe various interface mechanisms by which RDF Schema and XML Schema can be made to work together concurrently.

3. Combining RDF and XML Schemas

There are two possible alternative schema languages for defining application profiles (application-specific metadata element sets) : RDF Schema [5] and XML Schema [6,7,8]. (XML DTDs cannot seriously be considered as a solution since they do not explicitly support namespaces [21].) Of the two possible approaches, each offers its own advantages and disadvantages:

- RDF Schemas provide support for rich semantic descriptions but provide limited support for the specification of local usage constraints (i.e., structural, cardinality and datatyping constraints);
- XML Schemas provide support for explicit structural, cardinality and datatyping constraints but provide little support for the semantic knowledge necessary to enable flexible dynamic mapping between metadata domains.

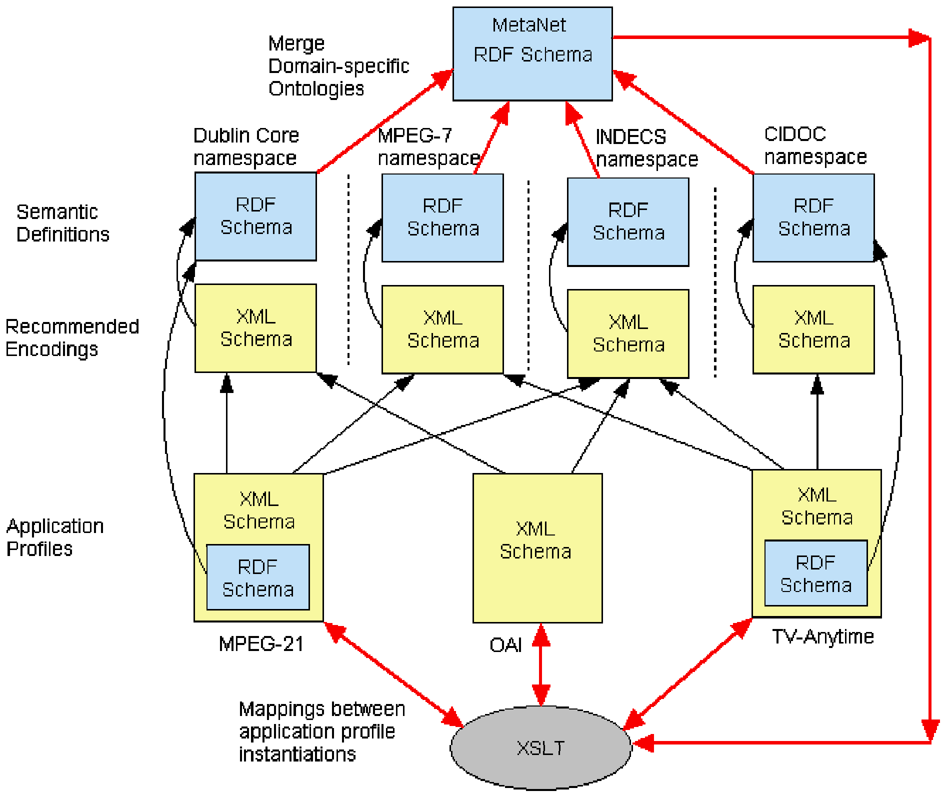


Figure 1 - Example of the Proposed Web Metadata Architecture

Hence the most logical approach is to use both RDF Schemas and XML Schemas so as to exploit their complementary features. The difficulties associated with using both schema languages in conjunction are that:

- there is a degree of functional overlap between RDF Schema and XML Schema which can be resolved by:
  - Either developing a hybrid RDF+XML schema parser which is capable of checking for consistency between RDF Schema and XML Schema constraints;
  - Or clearly demarcating the responsibilities of each schema language to prevent duplication or inconsistency between constraints;
- there are currently no clearly defined mechanisms for smoothly and cleanly meshing RDF Schema and XML Schema definitions.

In the long-term we believe that this calls for a re-examination of the two schema languages and the formulation of a design that integrates their complementary functionality. However, there is an immediate need for a more near-term solution to serve the critical need for metadata interoperability.

In the remainder of this section we propose various immediately-available solutions (and their advantages and disadvantages) to the problems outlined above which will enable RDF Schema and XML Schema to work in synergy to satisfy the requirements for metadata interoperability.

3.1 A Comparison of RDF Schema and XML Schema Representations

In this section we express a simple example in both XML Schema and RDF Schema to highlight the advantages and disadvantages of each schema language and to demonstrate the overlap in functionality.

Consider the following simple example: In our domain, we have a new class *Book* which is a subClassof *Resource*. The Book class has 2 properties, *title* and *author*. Each book may have one and only one title but may have up to four authors. Author is a subtype of the DCMES element *dc.creator* and also has an additional property of its own, *organisation*. The values of the *organisation* property are constrained to a set of three allowable instances ("OCLC," "Cornell University" and "DSTC").

Below is an RDF Schema representation for this example.

The RDF Schema *Class* and *Property* declarations and *label* and *comment* elements, provide semantic definitions for the metadata elements and their attributes. The type hierarchy is defined using the *subClassOf* and *subPropertyOf* elements. The *domain* constraint specifies the attachment of properties to classes and the *range* constraint can be used to indicate the classes that the values of a property must be members of.

```
<?xml version='1.0'?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"/>

<rdfs:Class rdf:ID="Book" >
   <rdfs:label>Book</rdfs:label>
   <rdfs:comment>The class of books</rdfs:comment>
   <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Resource"/>
</rdfs:Class>

<rdf:Property rdf:ID="title" >
  <rdfs:label>Title</rdfs:label>
  <rdfs:comment>The name given to the resource</rdfs:label>
  <rdfs:domain rdf:resource="#Book"/>
  <rdfs:range
     rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdfs:Property ID="author" >
  <rdfs:label>Author</rdfs:label>
  <rdfs:comment>A person responsible for creating a written
document</rdfs:comment>
  <rdfs:subPropertyOf
rdf:resource="http://purl.org/dc/elements/1.1/dcmes.rdf#Creator"/>
  <rdfs:domain rdf:resource="#Book"/>
  <rdfs:range
     rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdfs:Property>

<rdf:Property ID="organisation" >
  <rdfs:label>Organisation</rdfs:label>
  <rdfs:comment="The author's affiliation."/>
  <rdfs:domain rdf:resource="#Author"/>
  <rdfs:range rdf:resource="#OrgNames"/>
</rdf:Property>

<rdfs:Property rdf:ID="OrgNames"/>

<OrgNames rdf:ID="OCLC"/>
<OrgNames rdf:ID="Cornell University"/>
<OrgNames rdf:ID="DSTC"/>

</rdf:RDF>
```

Below is the corresponding XML Schema representation for the example above.

```
<schema xmnls="http://www.w3.org/1999/XMLSchema"
      targetNamespace="http://www.dstc.edu.au/"
      xmnls:dstc="http://www.dstc.edu.au/"
      xmlns:dc="http://purl.org/dc/elements/1.1/"/>

  <import namespace="http://purl.org/dc/elements/1.1/"/>

  <element name="Book" >
    <annotation>
      <documentation>The class of books</documentation>
    </annotation>

    <sequence>
      <element ref="dc:title" minOccurs="1" maxOccurs="1"/>
      <element name="author" type="author" maxOccurs="4"/>
    </sequence>
    <attribute name="id" type="uriReference"/>
  </element>

  <complexType name="author" >
     <extension base="dc:creator" >
       <element name="organisation" type="OrgNames"/>
      </extension>
  </complexType>

  <simpleType name="OrgNames" >
    <restriction base="string" >
      <enumeration value="OCLC"/>
      <enumeration value="Cornell University"/>
      <enumeration value="DSTC"/>
    </restriction>
  </simpleType>

</schema>
```

A comparison of the two schema language representations above, show that:

- RDF Schema provides support for richer semantic definitions through its ability to define type hierarchies, class/property relationships and human-readable descriptions (using the *label* and *comment* tags). But it provides limited support for the specification of local usage constraints (i.e., structural, cardinality and datatyping constraints);
- XML Schema provides support for explicit structural, cardinality and datatyping constraints but provides little semantic information.
- Overlap in functionality occurs in the following areas:
  - o Between the RDF Schema *range* constraint and XML Schema *type* constraints;
  - o Between the RDF Schema *domain* constraint and the content model definitions of XML Schema types and elements;
  - o In the definition of the enumerated list or controlled vocabulary values for organisations;
  - o Between RDF Schema *comments* and XML Schema *annotations*. Both provide human-readable descriptions of metadata elements or types.

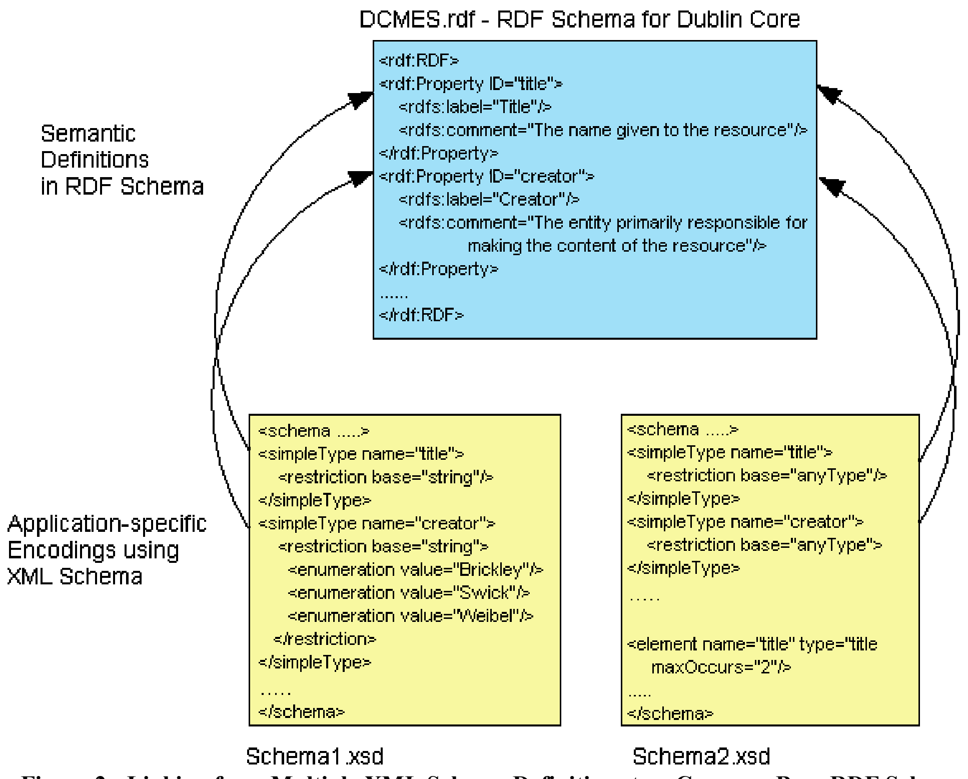
Since hybrid RDF+XML Schema validators, which are capable of validating both schemas and also checking for consistency between the two, don't yet exist, we need to clearly delineate the roles of the two schema languages to prevent duplication or inconsistencies between constraints.

For this reason we adopt the approach that the RDF Schema representation for a metadata element set should only contain semantic definitions. Because constraints on the attachment of properties to classes (*domain*) and property values (*range*) can also be expressed using XML Schema, we suggest that these particular RDF Schema constraints should not be used in this context and that such class/property relationship constraints and property value constraints should be expressed in the associated XML Schema file.

Similarly the XML Schema encroaches onto the semantic responsibilities which have been delegated to RDF Schema. When using both RDF Schema and XML Schema in conjunction, the XML Schema should contain only local usage constraints and no semantic definitions such as the semantic descriptions inside the *annotation* and *documentation* tags associated with each type.

3.2 Mechanisms for Interfacing RDF Schemas and XML Schemas

In section 3.1 we clarified the demarcation of responsibilities between RDF Schema and XML Schema when they are used in conjunction. In this section, we will now investigate how to link or


DCMES.rdf - RDF Schema for Dublin Core
Semantic Definitions in RDF Schema
Application-specific Encodings using XML Schema
Schema1.xsd
Schema2.xsd

```
<rdf:RDF>
<rdf:Property ID="title">
   <rdfs:label="Title"/>
   <rdfs:comment="The name given to the resource"/>
</rdf:Property>
<rdf:Property ID="creator">
   <rdfs:label="Creator"/>
   <rdfs:comment="The entity primarily responsible for
               making the content of the resource"/>
</rdf:Property>
......
</rdf:RDF>
```

```
<schema .....>
<simpleType name="title">
   <restriction base="string"/>
</simpleType>
<simpleType name="creator">
   <restriction base="string">
     <enumeration value="Brickley"/>
     <enumeration value="Swick"/>
     <enumeration value="Weibel"/>
   </restriction>
</simpleType>
.....
</schema>
```

```
<schema .....>
<simpleType name="title">
   <restriction base="anyType"/>
</simpleType>
<simpleType name="creator">
   <restriction base="anyType">
</simpleType>
.....

<element name="title" type="title
    maxOccurs="2"/>
.....
</schema>
```

Figure 2 - Linking from Multiple XML Schema Definitions to a Common Base RDF Schema

combine the two schema languages.
In section 3.1 we also demonstrated that RDF Schema is ideal for expressing the base semantic concepts for a particular domain's metadata model and XML Schema is ideal for expressing the local usage constraints (such as closed vocabularies, occurrence or formatting constraints). Because the underlying semantics will remain relatively stable compared to the syntax which will be application-dependent, we have chosen to make the RDF Schema the base schema and to point to the base RDF Schema from the application-specific XML Schemas. Figure 2 demonstrates the logic behind this approach.
In sections 3.2.1 and 3.2.2 we describe two methods for combining RDF Schema semantics with XML Schema local constraints:

1. Embedding the RDF Schema *Class/subClassOf, Property/subPropertyOf* definitions inside type annotations in the XML Schema file;
2. Adding links from the XML Schema to an external RDF Schema file.

3.2.1 Embedding Local RDF Schema Semantics in XML Schema Annotations

The first method involves incorporating local RDF Schema *Class*, *subClassOf*, *Property* and *subPropertyOf* definitions inside the XML Schema file. The only method for adding such extensions to XML Schema without loss of conformance is via the *annotation* and *appinfo* elements. *appinfo* appears as a subelement of *annotation* which may appear at the beginning of most schema constructions.
To illustrate, the following example shows how RDF semantics associated with the "title" and "creator" elements can be embedded in their corresponding type annotations in the XML Schema file. As suggested in Section 3.1, to prevent duplication or contradiction of constraints between the XML and RDF Schema definitions, the RDF Schema domain and range constraints have not been used.

```
<schema xmnls="http://www.w3.org/1999/XMLSchema"
   targetNamespace="http://purl.org/dc/elements/1.1/"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >

   <annotation>
     <documentation>
       Draft XML Schema for the Dublin Core Element Set, V 1.1
     </documentation>
   </annotation>

   <simpleType name="title" >
     <annotation>
       <appinfo>
         <rdf:Property ID="title" >
           <rdfs:label="Title" >
           <rdfs:comment="The name given to the resource." >
         </rdf:Property
       </appinfo>
     </annotation>
     <restriction base="string"/>
   </simpleType>
   <simpleType name="creator" >
     <annotation>
       <appinfo>
         <rdf:Property ID="creator" >
           <rdfs:label="Creator" >
           <rdfs:comment="An entity primarily responsible for
                          making the content of the resource" >
         </rdf:Property
       </appinfo>
     </annotation>
     <restriction base="string"/>
   </simpleType>
...
</schema>
```

Using this approach it is also possible to embed RDF Schema definitions of new classes or properties which are subClasses or subProperties of existing classes and properties defined in domain-specific RDF Schemas. For example:

```
<simpleType name="author" >
  <annotation>
    <appinfo>
      <rdf:Property ID="author" >
        <rdfs:subPropertyOf rdf:resource=
                    "http://purl.org/dc/elements/1.1/#Creator"/>
      </rdf:Property
    </appinfo>
  </annotation>
  <restriction base="string"/>
</simpleType>
```

This approach has the advantage of combining both the semantic definitions and structural and syntactic constraints in a single file, whilst maintaining XML Schema conformance. However, it is less flexible than the approach (shown in the next section) of separating the semantics of metadata elements from the usage constraints. This method also requires:

- either an RDF/XML Schema parser to be developed by adding extensions to an existing XML Schema parser (such as XSV [23]) to parse the embedded RDF-specific definitions;
- or an XSLT [24] program to extract the RDF Schema definitions into a separate RDF Schema file which can be parsed using an existing RDF Schema parser such as SiRPAC [25].

However, the major limitation of this approach is that those RDF classes and properties defined explicitly within XML Schema annotations are local definitions only and cannot be reused or pointed to by other schemas because they are not globally-accessible named elements. This conflicts with our reason for using RDF Schema which is to enable the dissemination and reuse of the semantic concepts across the Web to promote semantic interoperability, independent of the local usage constraints.

3.2.2 Linking External RDF Schema Definitions to an XML Schema

The second method involves using XLink [33] and the XLink Markup Name Control namespace proposed in a recent W3C Note[34], to link remote RDF Schema definitions in a separate file or namespace, to XML Schema type definitions.
In this Note, the authors suggest that an attribute of type *xl:arcrole*, defined in an XML Schema in the XLink namespace,

be added to each simple or complex type and that it be given a value that corresponds to an RDF property. This approach is illustrated in the example schema below. The problem with this approach is that the RDF semantics are only specified at time of instantiation not at the time of schema design, which is our requirement.

```
<schema xmnls="http://www.w3.org/1999/XMLSchema"
   targetNamespace="http://purl.org/dc/elements/1.1/"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:xl="http://www.w3.org/2000/10/xlink-ns" >

   <annotation>
     <documentation>
       Draft XML Schema for the Dublin Core Element Set, V 1.1
     </documentation>
   </annotation>

   <complexType name="title" >
     <simpleContent>
       <extension base="string" >
         <attribute name="arcrole" type="xl:arcrole"/>
       <extension>
     </simpleContent>
   </complexType>

   <complexType name="creator" >
     <simpleContent>
       <extension base="string" >
         <attribute name="arcrole" type="xl:arcrole"/>
       <extension>
     </simpleContent>
   </complexType>
...
</schema>
```

The corresponding instantiation would look something like:

```
<Description about="urn:isbn:0-65743-123-1" >
 <title arcrole="http://purl.org/dc/elements/1.1/dcmes.rdf#title">
      Where The Wild Things Are
   </title>
   <creator arcrole=
              "http://purl.org/dc/elements/1.1/dcmes.rdf#creator">
       Maurice Sendak
   </creator>
   .....

</Description>
```

A better approach is to specify a link to the type's corresponding semantics (RDF Property or Class definition) from within the XML Schema file. This is possible using the openness of XML Schema attributes. Since nearly all types are extended from the *openAttrs* type in the Schema for Schemas in [7], it is possible to extend XML Schema type definitions with a "semantics" attribute defined in another namespace. Using this approach, the value of the semantics attribute is the RDF Property or Class which defines the semantics of each simple or complex type.

We have chosen to link the semantics to XML Schema type definitions, rather than element declarations. This is because restrictions, extensions, redefinitions and elements are all built on top of XML Schema types, so the most logical and flexible approach is to attach the semantics to the *type* rather than the *element*.

```
<schema xmnls="http://www.w3.org/1999/XMLSchema"
  targetNamespace="http://purl.org/dc/elements/1.1/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:xx="http://www.example.org/XMLRDFSchemaBridge" >

   <annotation>
     <documentation>
       Draft XML Schema for the Dublin Core Element Set, V 1.1
     </documentation>
   </annotation>

   <simpleType name="title"  xx:semantics=
          "http://purl.org/dc/elements/1.1/dcmes.rdf#title">
     <restriction base="string"/>
   </simpleType>

   <simpleType name="creator" xx:semantics=
          "http://purl.org/dc/elements/1.1/dcmes.rdf#creator">
     <restriction base="string"/>
   </simpleType>
...
</schema>
```

4. MetaNet - A Common Ontology for Semantic Interoperability

Semantic knowledge in the form of an ontology or thesaurus is required to enable flexible, dynamic mapping between XML-encoded instantiations of application profiles. Since this semantic information is already available in the separate RDF Schemas provided by each domain, the task remains to merge these RDF Schemas into a single RDF Schema representation of the merged ontologies and to link this to XSLT programs to perform dynamic mappings between metadata descriptions.

In this section we describe a metadata thesaurus, MetaNet, which has been generated by merging a number of domain-specific vocabularies manually. Ideally, this would be machine-generated using inferencing, such as has been proposed in the Ontology Inference Layer (OIL) [26].

MetaNet [27] is a thesaurus which contains preferred terms, equivalent/overlapping terms (ET), narrower terms (NT) and broader terms (BT) which encompass most of the significant metadata models/vocabularies/standards. The top-level preferred terms are based on the core ABC vocabulary developed by the Harmony project [28, 29].

The objective of the *MetaNet* thesaurus is to provide the semantic knowledge required to enable machine understanding of equivalence and hierarchical (subtyping) relationships between metadata terms from different domains. The scope of this thesaurus is limited to the most significant metadata models/vocabularies/standards used for describing attributes and events associated with resources and their life cycles. This encompasses metadata vocabularies from the bibliographic, museum, archival, record keeping and rights management communities. It has been developed by performing WordNet [30] searches using the core terms from the ABC vocabulary and extracting those synonyms and hyponyms which could conceivably be used in a metadata scheme to represent the original core term. In addition the majority of metadata terms from the vocabularies of the DC, INDECS, IFLA and CIDOC CRM have been manually incorporated into the thesaurus.

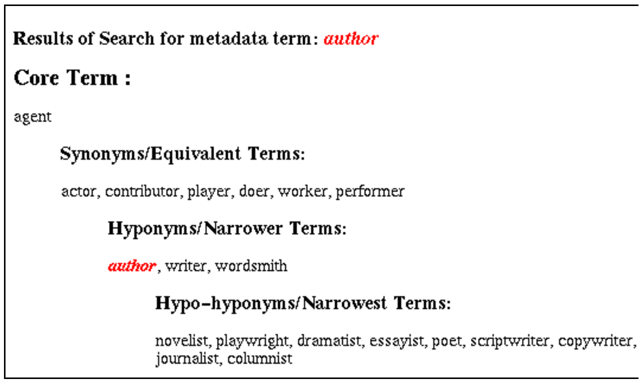
For example, consider "Agent" which is a core entity of the ABC model and a core term of the ABC vocabulary [29].

Semantically equivalent terms for "Agent" which are used within other metadata vocabularies include: *actor, contributor, player, doer, worker, performer* Possible narrower terms or *hyponyms* for "Agent" include: *creator, author, composer, artist, musician, etc.*.

An RDF Schema representation of this thesaurus has been developed. The RDF and RDF Schema elements, *Class*, *subClassOf*, *Property*, *subPropertyOf* are used to define the type hierarchy and entity/attribute relationships between metadata elements. The RDFS *label* element is used to specify terms which are considered to be semantically equivalent. Below is an excerpt from the RDF Schema which illustrates the representation for the "Agent" metadata term as well as its equivalent terms and a partial hierarchy of its narrower terms.

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

<rdfs:Class rdf:ID="Agent" >

   <rdfs:comment  xml:lang="en" >The resources which
contribute to or act in an event. Typically agents are people,
groups of people, organisations or instruments.</rdfs:comment>

   <rdfs:label xml:lang="en" >Actor</rdfs:label>
   <rdfs:label xml:lang="en" >Contributor</rdfs:label>
   <rdfs:label xml:lang="en" >Player</rdfs:label>
   <rdfs:label xml:lang="en" >Doer</rdfs:label>
   <rdfs:label xml:lang="en" >Worker</rdfs:label>
   <rdfs:label xml:lang="en" >Performer</rdfs:label>

   <rdfs:subClassOf  rdf:resource=
        "http://www.w3.org/2000/01/rdf-schema#Resource"/>

</rdfs:Class>

<rdfs:Class rdf:ID="Author" >
   <rdfs:label xml:lang="en" >Writer</rdfs:label>
   <rdfs:label xml:lang="en" >Wordsmith</rdfs:label>
   <rdfs:subClassOf  rdf:resource="#Agent"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Journalist" >
   <rdfs:label xml:lang="en" >Columnist</rdfs:label>
   <rdfs:label xml:lang="en" >Reporter</rdfs:label>
   <rdfs:subClassOf rdf:resource="#Author"/>
</rdfs:Class>
</rdf:RDF>
```

A web search and browse interface to MetaNet has also been developed [27]. Users can search on any common metadata term and retrieve a list of equivalent terms, broader terms and narrower terms. Figure 3 shows the results of a search on the term "author."



Results of Search for metadata term: *author*

Core Term :

agent

Synonyms/Equivalent Terms:

actor, contributor, player, doer, worker, performer

Hyponyms/Narrower Terms:

author, writer, wordsmith

Hypo-hyponyms/Narrowest Terms:

novelist, playwright, dramatist, essayist, poet, scriptwriter, copywriter, journalist, columnist

Figure 3 - Results of MetaNet Search

In the next section, we describe mechanisms by which XSLT can access the semantic knowledge held in the MetaNet RDF Schema to perform the semantic mapping component of metadata description transformations.

5. Adding Semantic Knowledge to XSLT

The Extensible Style Transformation Language's (XSLT) [24] ability to transform data from one XML representation to another appears to makes it ideal for metadata interchange applications.

In order to evaluate XSLT's capabilities for mapping between application profile instantiations, we generated two hybrid schemas (which use both XML Schema and RDF Schema) and then attempted to map between instantiations of these schemas using XSLT.

Table 1 below shows the two application profile examples. Using XSLT and the Xalan [32] XSLT processor we developed XSL programs for transforming from myDescription1 to myDescription2.

Application Profile Examples

```
<schema xmnls="http://www.w3.org/1999/XMLSchema"
  targetNamespace="http://www.dstc.edu.au"
  xmnls:dstc="http://www.dstc.edu.au"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:mpeg7="http://www.mpeg.org/MPEG7/2000/"
  xmlns:ims="http://ltsc.ieee.org/doc/wg12/"/>

  <import namespace="http://purl.org/dc/elements/1.1/"/>
  <import namespace="http://www.mpeg.org/MPEG7/2000/"/>
  <import namespace=""http://ltsc.ieee.org/doc/wg12/"/>

  <element name="myDescription1" >
    <complexType>
      <sequence>
        <element ref="dc:title" minOccurs="1" maxOccurs="2" >
        <element ref="dc:creator" minOccurs="1" maxOccurs="3" >
        <element ref="mpeg7:UsageMetaInformation"
                    minOccurs="0" maxOccurs="unbounded"/>
        <element ref="ims:LearningContext"/>
      </sequence>
    </complexType>
    <attribute name="about" type="uriReference"/>
  </element>

</schema>
```

```
<schema xmnls="http://www.w3.org/1999/XMLSchema"
      targetNamespace="http://www.dstc.edu.au/"
      xmnls:dstc="http://www.dstc.edu.au"
      xmnls:dc="http://purl.org/dc/elements/1.1/"
      xmlns:ims="http://ltsc.ieee.org/doc/wg12/"/>

  <import namespace="http://purl.org/dc/elements/1.1/"/>
  <import namespace=""http://ltsc.ieee.org/doc/wg12/"/>

  <element name="myDescription2" >
    <element ref="ims:title" minOccurs="1" maxOccurs="1"/>
    <element name="author" type="author"/>
    <element ref="dc:rights"/>
    <element ref="ims:TypicalAgeRange"/>
    <attribute name="about" type="uriReference"/>
  </element>

  <complexType name="author" >
    <annotation>
      <appinfo>
        <rdf:Property ID="author" >
          <rdfs:subPropertyOf
             rdf:resource="http://purl.org/dc/elements/1.1/#Creator"/>
        </rdf:Property
      </appinfo>
    </annotation>
    <sequence>
      <element ref="dc:creator"/>
      <element name="organisation" type="OrgNames"/>
    </sequence>
  </complexType>

  <simpleType name="OrgNames" >
    <restriction base="string" >
      <enumeration value="OCLC"/>
      <enumeration value="University of Cambridge"/>
      <enumeration value="DSTC"/>
    </restriction>
  </simpleType>

</schema>
```

The mapping implementations revealed that XSLT is inadequate for implementing flexible dynamic semantic mappings between metadata vocabularies. This is due to:

- XSLT's limited capabilities for handling variable input descriptions based on schemas which are not tightly constrained;
- The non-existence of machine-understandable semantic information in declarative XML-encoded metadata descriptions;
- Processor-dependent handling of input parameters and procedural code extensions;
- Limited string manipulation and comparison functions, e.g., it is not possible to perform case-insensitive string comparisons within XSLT.

A previous paper by Alison Cawsey which investigated the use of XSLT for customizing RDF descriptions, reached similar conclusions [31].

Semantic knowledge in the form of an ontology or thesaurus is required to enable flexible, dynamic mapping between XML-encoded metadata descriptions. This semantic information is available already in the MetaNet thesaurus (described in Section 4) which was generated by merging domain-specific ontologies. Hence we needed to determine a method to link the semantic information in MetaNet to the XSLT program performing the mapppings.

Using XSLT, it is possible to parse an input XML description and for each new element encountered, call a Java procedural code extension which determines the equivalent term in the output domain from the MetaNet thesaurus. For example, suppose the Java program, Mapping.java, contains a *readMetaNet* function. For each element encountered during parsing, the input element name (e.g., 'dstc:Author') and the output domain (e.g., 'dc') are passed to the *readMetaNet* function. This function searches the MetaNet RDF Schema file for the equivalent output domain element (e.g., dc:creator), returns this value and XSL creates a new output element with this name in the output description. Figure 4 below illustrates the program flowchart.

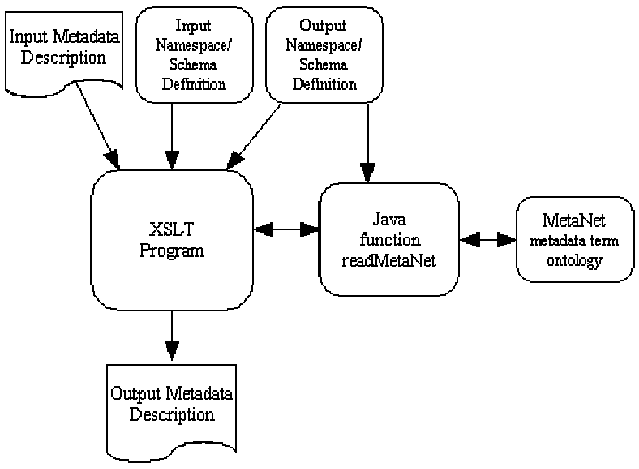

Input Metadata Description
Input Namespace/ Schema Definition
Output Namespace/ Schema Definition
XSLT Program
Java function readMetaNet
MetaNet metadata term ontology
Output Metadata Description

Figure 4 - Program Flow for Metadata Description Mappings

The XSL code below illustrates how to call a Java program function, readMetaNet, from the main XSL file.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns:dc ="http://purl.org/dc/elements/1.1/" >
     xmlns:lxslt="http://xml.apache.org/xslt"
     xmlns:mapping="Mapping"
     extension-element-prefixes="mapping"
     version="1.0" >

  <lxslt:component prefix="mapping" elements="*"
                     functions="readMetaNet" >
     <lxslt:script lang="javaclass" src="Mapping"/>
  </lxslt:component>
  <xsl:template match="*" >
    <xsl:element name="mapping:readMetaNet(., 'dc')"/>
      <xsl:value-of select="."/>
    </xsl:element>
  </xsl:template>

</xsl:stylesheet>
```

Below is a high-level simplistic algorithm describing the mapping process which is performed within the readMetaNet Java function in Figure 4:

```
For each element in the input description
{
  Search for the input element name in the output domain schema;
  if (found) {
    Map the input element to the equivalent output domain element;
  {
  else {
    Extract the Equivalent Terms (ETs) to the input element from MetaNet;
    Search the output domain schema for each of the ETs;
    if (an ET is found)
    {
      Map the input element to the equivalent output domain element;
    }
    else {
      Extract the broader terms (BTs) for the input element from MetaNet;
      Search for each BT in the output domain namespace;
      if (a BT is found)
      {
        Map the input element to the broader output domain element;
      }
      else {
        Extract the narrower terms (NTs) for the input element from MetaNet;
        Search for each NT in the output domain namespace;
        if (a NT is found)
        {
          Map the input element to the narrower output domain element;
        }
      }
    }
  }
} endFor
```

By adding a procedural code extension to XSLT to perform the semantic mapping (using information on semantic relationships between metadata terms in MetaNet), we are able to execute dynamic, flexible mappings between XML-encoded instantiations of application profiles.

6. Conclusions

In this paper, we have proposed a web metadata architecture which combines the best features of both XML Schema and RDF Schema to enhance metadata interoperability across the web.

XML Schemas are used for their ability to explicitly define local usage constraints such as content model, occurrence and datatyping constraints. These features make XML Schema language ideal for defining application profiles. RDF Schemas are used to express the semantics of domain-specific metadata models in a machine-understandable syntax which can be used to merge ontologies from multiple domains.

We have suggested approaches for combining XML Schemas and RDF Schemas based on the currently available mechanisms. The overlap in functionality between these two schema languages and the lack of clearly defined mechanisms or tools for linking RDF Schemas and XML Schemas have made this task difficult and the available solutions cumbersome. For example, development of a hybrid RDF+XML Schema parser to check for consistency of constraints between two corresponding schemas for the one underlying model would be extremely useful.

Ideally the XML Schema language would provide an explicit built-in attribute on simple or complex types, which is a uriReference to the corresponding semantics for that type, i.e., existing classes or properties in an external RDF Schema. This would preclude the need for *semantics* attribute definition in the XMLRDFSchemaBridge namespace. For example:

```
<simpleType name="originator" semantics=
          "http://purl.org/dc/elements/1.1/dcmes.rdf#creator"/>
    <restriction base="string"/>
</simpleType>
```

This work has also shown that the current extensibility mechanisms for both XML Schema and RDF Schema are unclear and require clarification, simplification and implementation examples.

We have also described MetaNet, a generic metadata term thesaurus, expressed in RDF Schema which was generated by manually merging RDF Schemas from different metadata domains. In addition, we have shown how the semantic knowledge in the MetaNet thesaurus can be accessed by a procedural code extension to XSLT to enable flexible, dynamic mappings between application profile instantiations.

In the future we are interested in investigating the application of more lightweight rules-based approaches such as Schematron [35] in combination with RDF Schema to support interoperable application profiles.

Our final conclusion is that although we have demonstrated how each of these web metadata architectural components can be made to fit together, the process has been analogous to the assembly of a badly made jigsaw puzzle. The joins have not been intuitive, clean or easy and some parts are missing all together. Based on the work described in this paper, we suggest that before either schema language moves to the Proposed Recommendation or Recommendation stage, there is a need for a re-examination of the two schema languages and the formulation of mechanisms which cleanly and smoothly integrate their complementary functionality.

Acknowledgements

The work described in this paper has been carried out as part of the Harmony Project. It has been funded by the Cooperative Research Centre for Enterprise Distributed Systems Technology (DSTC) through the Australian Federal Government's CRC Programme (Department of Industry, Science and Resources) and NSF Grant 9905955. The authors also wish to acknowledge the valuable contribution which discussions with Dan Brickley have made to this paper.

References

[1] R. Heery, M. Patel, "Application Profiles: mixing and matching metadata schemas", Ariadne Issue 25, September 2000. <http://www.ariadne.ac.uk/issue25/app-profiles/>

[2] TV-Anytime Forum, <http://www.tv-anytime.org/>

[3] MPEG-21 Multimedia Framework, <http://www.cselt.it/mpeg/public/mpeg-21_pdtr.zip>

[4] Open Archives Initiative. <http://www.openarchives.org/>

[5] RDF Schema Specification 1.0, W3C Candidate Recommendation 27 March 2000. <http://www.w3.org/TR/rdf-schema/>

[6] XML Schema Part 0: Primer, W3C Candidate Recommendation, 24 October 2000, <http://www.w3.org/TR/xmlschema-0>

[7] XML Schema Part 1: Structures, W3C Candidate Recommendation, 24 October 2000, <http://www.w3.org/TR/xmlschema-1>

[8] XML Schema Part 2: Datatypes, W3C Candidate Recommendation, 24 October 2000, <http://www.w3.org/TR/xmlschema-2>

[9] The SCHEMAS Project, Forum for Metadata Schema Implementers, <http://www.schemas-forum.org/>

[10] J. Hunter, An XML Schema Approach to Application Profiles, October 3 2000. <http://archive.dstc.edu.au/maenad/appln_profiles.html>

[11] T. Berners-Lee, "XML and the Web", XML World, September, 2000. <http://www.w3.org/2000/Talks/0906-xmlweb-tbl/Overview.html>

[12]The Cambridge Communiqué, W3C Note 7 October 1999. <http://www.w3.org/TR/schema-arch>

[13]The Dublin Core Metadata Initiative. <http://www.purl.org/dc/>

[14] The BIBLINK Core Application Profile. <http://www.schemas-forum.org/registry/schemas/biblink/BC-schema.html>

[15] G. Rust, M. Bide, "The indecs Metadata Schema Building Blocks," Indecs Metadata Model, November, 1999. <http://www.indecs.org/results/model.htm>

[16] MPEG-7 Home Page <http://www.darmstadt.gmd.de/mobile/MPEG7/index.html/>

[17] Content Standard for Digital Geospatial Metadata (CSDGM), <http://www.fgdc.gov/metadata/contstan.html>

[18] The Gateway to Educational Materials <http://www.the gateway.org>

[19] IEEE Learning Technology Standards Committee's Learning Object Meta-data Working Group. Version 3.5 Learning Object Meta-data Scheme.

[20] ICOM/CIDOC Documentation Standards Group, Revised Definition of the CIDOC Conceptual Reference Model, September 1999. <http://www.geneva-city.ch:80/musinfo/cidoc/oomodel>

[21] Namespaces in XML, W3C Recommendation 14 January, 1999. <http://www.w3.org/TR/REC-xml-names>

[22] Dublin Core Metadata Element Set, Version 1.1, 2 July, 1999. <http://www.purl.org/dc/documents/rec-dces-19990702.htm>

[23] Validator for XML Schema, 22 September 2000 version. <http://www.w3.org/2000/09/webdata/xsv>

[24] XSL Transformations (XSLT) Version 1.0 W3C Recommendation 16 November 1999, <http://www.w3.org/TR/xslt.html>

[25] SiRPAC, Simple RDF Parser and Compiler. <http://www.w3.org/RDF/Implementations/SiRPAC>

[26] Ontology Inference Layer, <http://www.ontoknowledge.org/oil/>

[27] MetaNet Search Page, <http://sunspot.dstc.edu.au:8888/Metanet/Top.html>

[28] The Harmony Project Home Page, <http://www.ilrt.bris.ac.uk/discovery/harmony/>

[29] C.Lagoze, J. Hunter, D. Brickley, "An Event-Aware Model for Metadata Interoperability," ECDL 2000, Lisbon, September 2000.

[30] WordNet - a Lexical Database for English. <http://www.cogsci.princeton.edu/~wn/online/>

[31] A. Cawsey, "Presenting tailored resource descriptions: Will XSLT do the job?", WWW9, Amsterdam, May 2000. <http://www.cee.hw.ac.uk/~alison/www9/paper.html>

[32] Xalan-Java Overview. <http://xml.apache.org/xalan/overview.html>

[33] XML Linking Language (XLink) Version 1.0, W3C Candidate Recommendation, 3 July 2000. <http://www.w3.org/TR/xlink/>

[34] XLink Markup Name Control W3C Note 24 October 2000, <http://www.w3.org/TR/xlink-naming>

[35]The Schematron - An XML Structure Validation Language using Patterns in Trees. <http://www.ascc.net/xml/resource/schematron>